ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ ГАГАРИНА Ю.А.»

Институт прикладных информационных технологий и коммуникаций

Кафедра «Прикладные информационные технологии»

Направление 09.03.03 «Прикладная информатика»

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

«Разработка геоинформационной системы для маломобильных групп населения»

Студент (ка) Малышева Елена Сергеевна
фамилия, имя, отчество

курс 4 группа б2-ПИНФ41

Руководитель
к.т.н., доцент каф. ПИТ 11.06.20 Шварц А.Ю.
должность, ученая степень, уч. звание подпись, дата Инициалы Фамилия

Допущен к защите
Протокол № 12 от «11» июня 2020 года

Зав. кафедрой «Прикладные информационные технологии»

кандидат технических
наук, доцент 11.06.20 О.А. Торопова
ученая степень, уч. звание подпись, дата Инициалы Фамилия

Саратов 2020г

## Введение

Доступная среда — совокупность мер, включающих в себя оборудование различных объектов городской инфраструктуры, которые помогают маломобильным гражданам лучше ориентироваться в пространстве, свободнее передвигаться по улице или в помещениях, быстрее адаптироваться к самостоятельной жизни.

Маломобильными гражданами здесь называются люди, в силу различных обстоятельств испытывающие трудности в самостоятельном передвижении по городу и внутри зданий. Несмотря на старания властей, в ряде мест на территории Саратовской области всё ещё наблюдаются проблемы с доступностью многих объектов городской инфраструктуры, например, отсутствие пандусов, лифтов, ровного дорожного покрытия. Одним из аспектов решения данной проблемы может стать использование геоинформационной системы для сбора и предоставления по запросу информации о доступности объектов. Создание такой системы позволит пользователю заранее выбирать удобные для посещения городские объекты, чтобы физически и психологически стать более независимым и мобильным.

Целью выпускной квалификационной работы является разработка геоинформационной системы для маломобильных групп населения. Для достижения данной цели необходимо решить следующие задачи:

- Провести анализ предметной области;
- Провести проектирование геоинформационной системы;
- Выбрать инструменты разработки;
- Разработать базу данных и модули геоинформационной системы;
- Провести проверку работоспособности системы.

# 1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1. Геоинформационные системы

Геоинформационные системы (ГИС) [1] – это передовые компьютерные технологии, разработанные для создания карт и дополнения их различными данными: информацией об объектах и событиях, маршрутах и т.д. При этом стандартные операции с базами данных (введение сведений, извлечение статистических результатов) дополняются визуализацией и пространственным обзором.

Благодаря описанным характеристикам геоинформационные системы активно применяются для решения большого количества задач:

- Социологические исследования;
- Планирование перспективных решений в градостроительстве;
- Оценка результативности предпринимательской деятельности, маркетинговых стратегий;
- Анализ экологических проблем: водоёмов, изменения климата, загрязнение и изменение береговых линий водоёмов;
- Задачи логистики.

Помимо глобальных целей, с помощью данного вида программного обеспечения возможно регулирование локальных вопросов, например:

- Нахождение оптимального маршрута между точками;
- Выбор удобного расположения для фирмы;
- Поиск нужного здания по адресу;
- Муниципальные задачи.

Несмотря на то, что географический анализ появился достаточно давно, геоинформационные системы являются современным, эффективным и удобным для конечного пользователя инструментом, автоматизирующим процесс сбора данных, их обработки и наглядного представления.

Сейчас геоинформационные системы – востребованная область бизнеса, в ней заняты миллионы специалистов со всего мира. В России

существует более двухсот компаний, разрабатывающих и внедряющих ГИС во все сферы жизни, например, «Интегро (Альбея)» в Уфе, или «Трисофт» в Москве.

### 1.1.1 Структура ГИС

Структура ГИС представлена на рис. 1.

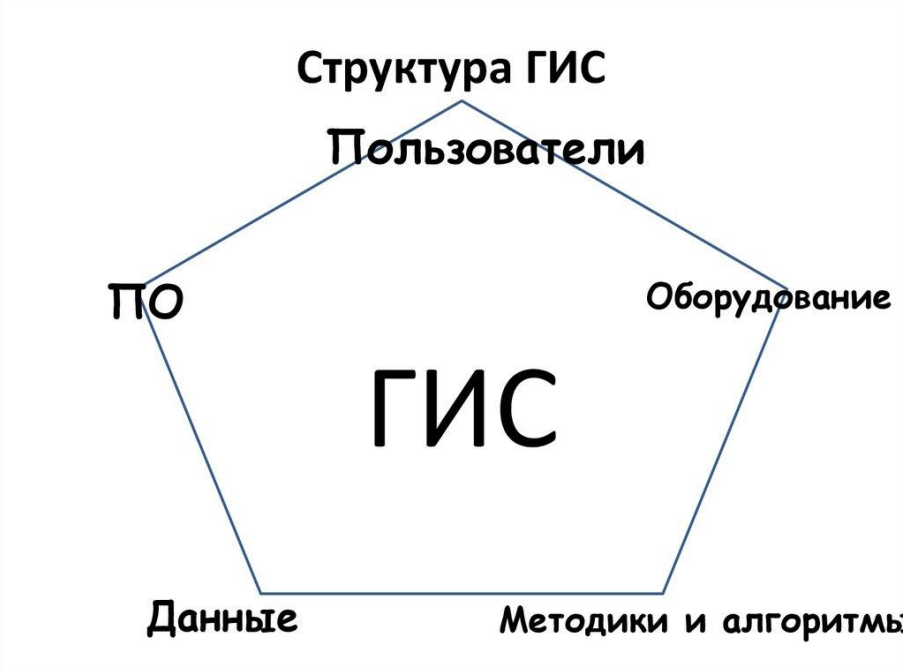


Рисунок 1 – Схема структуры ГИС

Приведем описание основных составляющих ГИС [2].

Оборудование – множество различных компьютерных платформ, от персональных компьютеров до крупных централизованных серверов.

Программное обеспечение – инструментарий для получения, обработки и визуализации информации. Отдельно стоит выделить программные модули для:

- введения данных и манипулирования данными;
- управления базой данных (СУБД);
- отображения пространственных запросов;
- доступа (пользовательский интерфейс).

Данные – информация о местоположении объектов на карте, адреса, описания, фотографии мест. Ради хранения, доступа и манипулирования этими данными и создаётся система.

Пользователи – разработчики, модераторы, администраторы сервиса, отвечающие за его корректное функционирование, а также обычные люди, которые решили воспользоваться системой в личных целях.

Методики и алгоритмы выбираются исходя из особенностей предметной области, где будет применяться геоинформационная система, например, ГИС для маломобильных граждан должна учитывать и визуализировать не только расположение здания, но и его оснащение для комфортного пребывания определённой группы населения [3]: пандусы, тактильная плитка, звуковое сопровождение и т.д.

### 1.1.2 Виды ГИС

Классификация географических информационных систем осуществляется по принципу охвата территории:

1. Глобальные (национальные и субконтинентальные) – дают возможность оценить ситуацию в масштабах планеты. Благодаря этому можно спрогнозировать и предотвратить природные и техногенные катаклизмы, оценить размер бедствия, спланировать ликвидацию последствий и организацию гуманитарной помощи. Применяются во всем мире с 1997 года.

Одна из самых известных глобальных ГИС Google Earth, разработанная компанией Google, позволяет делать снимки всей поверхности Земли в отличном разрешении.


Рисунок 2 – Google Earth

2. Региональные (локальные, субрегиональные, местные) – действуют на муниципальном уровне. Такие технологии отражают многие ключевые ниши: инвестиции, имущество, навигация, обеспечение безопасности населения и другие. Они помогают предпринимать шаги для развития определенного района, что способствует привлечению к нему капитала и экономическому росту. Примером региональной ГИС может служить сервис для визуализации охотничьих угодий в Саратовской области.

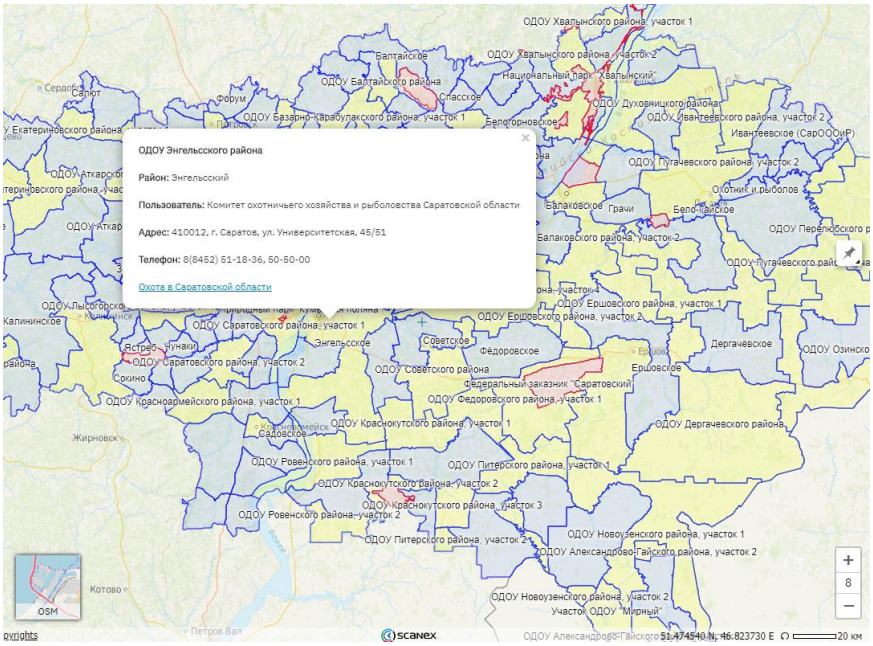


Рисунок 3 – Геопортал «Карта охотника»

### 1.1.3 Функционирование системы

Фактическая информация об объектах в ГИС хранит в виде подборки, объединённых по принципу географического положения тематических слоёв. Подобный подход позволяет решать разнообразные вопросы по реорганизации местности и проведению мероприятий.

Для того, чтобы определить местоположение объекта используются координаты точки, её адрес, индекс, номер дома, земельного участка, и т.д. Нужная информация появляется на карте после процедуры геокодирования.

Технологии применимы к растровым и векторным моделям.

Объект в векторной форме кодируется и сохраняется в виде набора координат. Такая форма больше подходит для устойчивых элементов с неизменными свойствами: реками, трубопроводами, полигонами.

Растровая схема имеет блоки сведений об отдельных составляющих. Она удобна для работы с переменными характеристиками, например, типами почв и доступностью объектов.

### 1.1.4 Применение ГИС

Возможности геоинформационной системы сегодня востребованы во всех сферах жизни. Наибольшее распространение они получили в следующих областях:

1. Землеустройство. Утилиты нужны для составления кадастров, вычисление площадей элементов, разметка границ земельных участков.

2. Управление размещением объектов. Здесь применение актуально для построения архитектурного плана, согласование сети промышленных, торговых и других точек специального назначения [4].

3. Районное развитие. Инженерные изыскания конкретных мест, решения задач по оптимизации инфраструктуры и привлечению инвесторов в настоящее время невозможны без детального изучения с помощью подобных структур.

4. Охрана природы. Геоинформационные системы позволяют проводить экологический мониторинг, планирование использования ресурсов.

5. Прогнозирование ЧС. Отслеживание изменений в разных геологических состояниях позволяет предсказать вероятность катастроф, разрабатывать меры для предотвращения и минимизации потерь от них.

6. Социальная навигация. Применение геоинформационных систем в этой области - визуализация на карте специально оборудованных для маломобильных граждан мест и построение оптимальных маршрутов существенно облегчает им перемещение по местности [5-8].

## 1.2 Анализ существующего программного обеспечения в данной области

### 1.2.1 ГИС «Доступная среда»

Прототипом разрабатываемой геоинформационной системы является ГИС «Доступная среда» [9], разработанная правительством Самарской области.

ГИС «Доступная среда» – геоинформационная система для учёта доступности социальной инфраструктуры.

Система имеет интуитивно понятный интерфейс, объекты отсортированы по доступности для разных категорий инвалидности: нарушения опорно-двигательного аппарата, зрения, слуха и ментальные. Также добавлены индикаторы доступности объектов: зелёный – «доступно», жёлтый – «частично доступно», красный – «недоступно». Регистрироваться в системе могут только члены правительства и работники государственных учреждений. Пользователю доступен только просмотр добавленных на карту сведений.

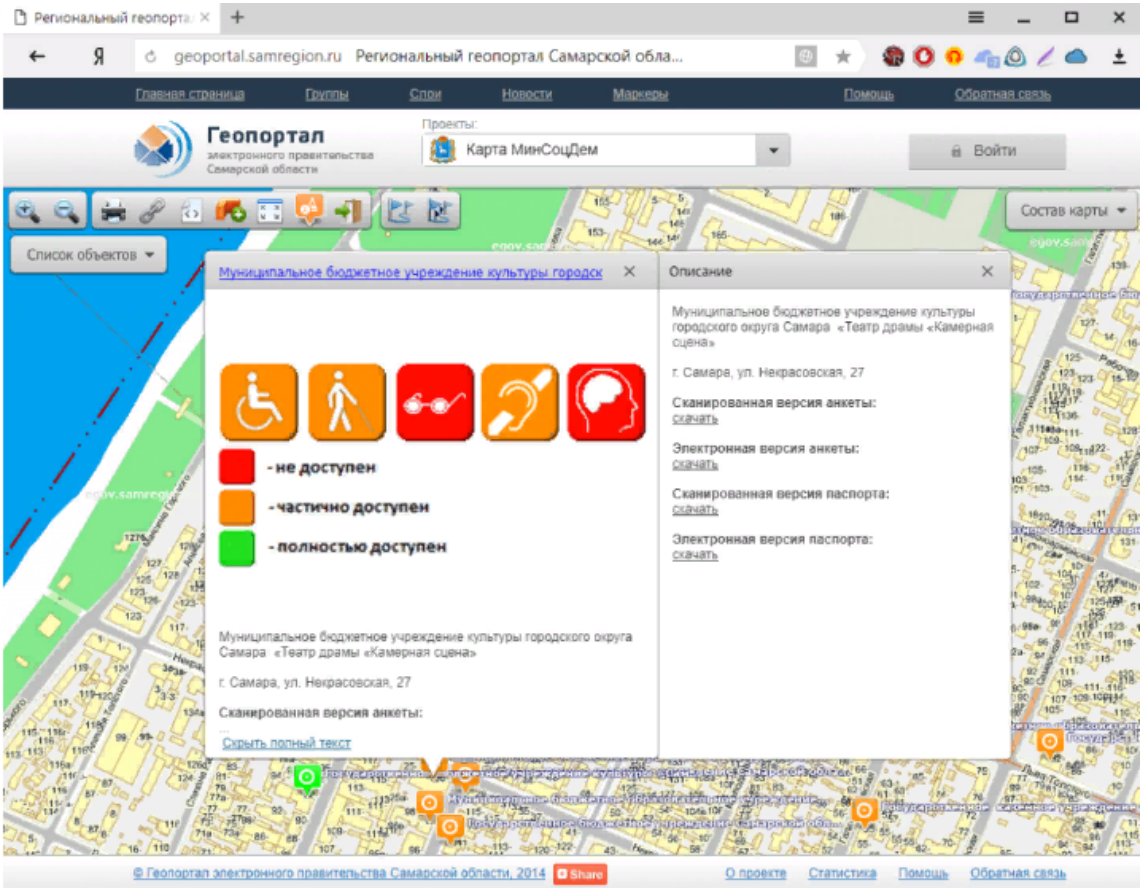


Рисунок 4 – Интерфейс ГИС «Доступная среда»

ГИС Доступная среда разрабатывалась с целью решения основных проблем, связанных с информационным обеспечением безбарьерного доступа маломобильных групп населения к транспортным, инженерным и социальным объектам городской инфраструктуры.

Система решает вопросы создания единой электронной базы данных об объектах социальной инфраструктуры и их доступности, поддержки процесса паспортизации объектов социальной инфраструктуры, создание единой информационной среды для взаимодействия с участниками процесса паспортизации объектов, реализации функций учета и контроля за созданием безбарьерной среды, формирования отчетных документов, информирование населения о доступности объектов социальной инфраструктуры, отслеживание сообщений о проблемах доступности объектов социальной инфраструктуры.

Приведем описание основных компонентов ГИС «Доступная среда».

Основным компонентом является учётная система.

Это распределенный веб-компонент для ведения реестра объектов социальной инфраструктуры, интегрированный с региональным геопорталом Самарской области для определения местоположения объектов.

Пользователями этой системы являются сотрудники министерств, представители различных исполнительных органов, представители экспертных комиссий и сотрудники контролирующих организаций.

В рамках системы учета дополняется информация об объектах социальной инфраструктуры, юридических лицах (собственниках) и экспертных комиссиях. Также в рамках системы доступны для скачивания законодательные акты, методы и шаблоны отчетных документов.

Для каждого объекта социальной инфраструктуры в рамках системы учета заполняются общие сведения, информация об организации, информация о характеристиках объекта и объеме предоставляемых услуг. Объекты социальной инфраструктуры привязаны к картографической системе регионального геопортала Самарской области. В рамках системы бухгалтерского учета также разработаны электронные варианты заполнения анкет и паспортов, в соответствии с которыми создаются соответствующие отчетные документы.

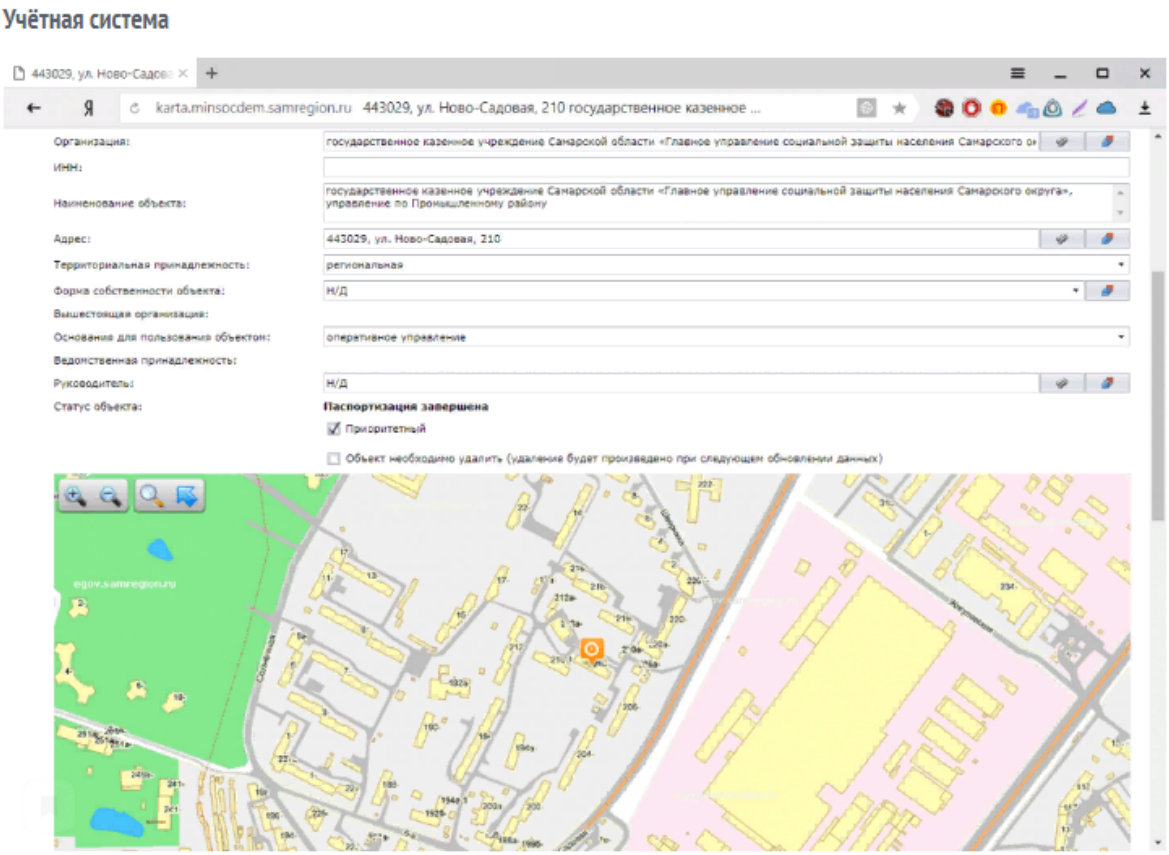


Рисунок 5 – Учётная система ГИС «Доступная среда»

Геопортал используется для информирования общественности о наличии объектов социальной инфраструктуры. На региональном геопортале Самарской области создан проект Министерства социальной и демографической политики.

Объекты представлены метками на карте. Пользователи могут перемещать и масштабировать карту для удобного поиска объектов. Для каждого объекта на геопортале есть описание, содержащее адрес, название организации, ссылки на скачивание отсканированных версий анкеты и паспорт. Существует также множество фотографий объекта, разбитых на функциональные зоны.

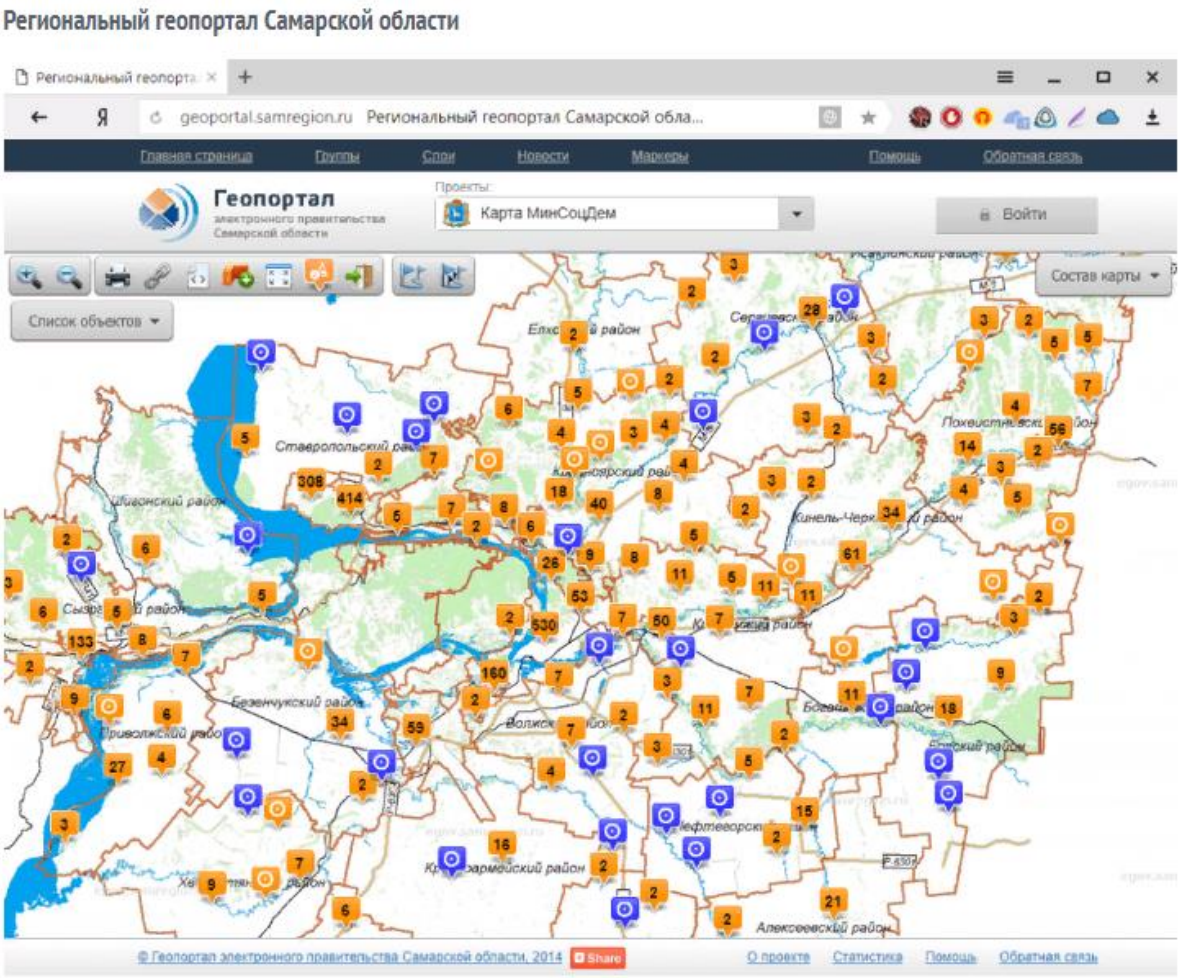


Рисунок 6 – Геопортал ГИС «Доступная среда»

Для операционной системы Android было разработано мобильное приложение с возможностью просмотра и редактирования информации об объектах социальной инфраструктуры. Оно позволяет:

1. Просматривать и редактировать характеристики объектов социальной инфраструктуры, в том числе их расположение на карте.

2. Просматривать и редактировать профили и паспорта объектов.

3. Приложение фотографий.

Редактирование данных возможно только для авторизованных пользователей. Чтобы получить доступ к редактированию данных, необходимо зарегистрироваться в системе учета.

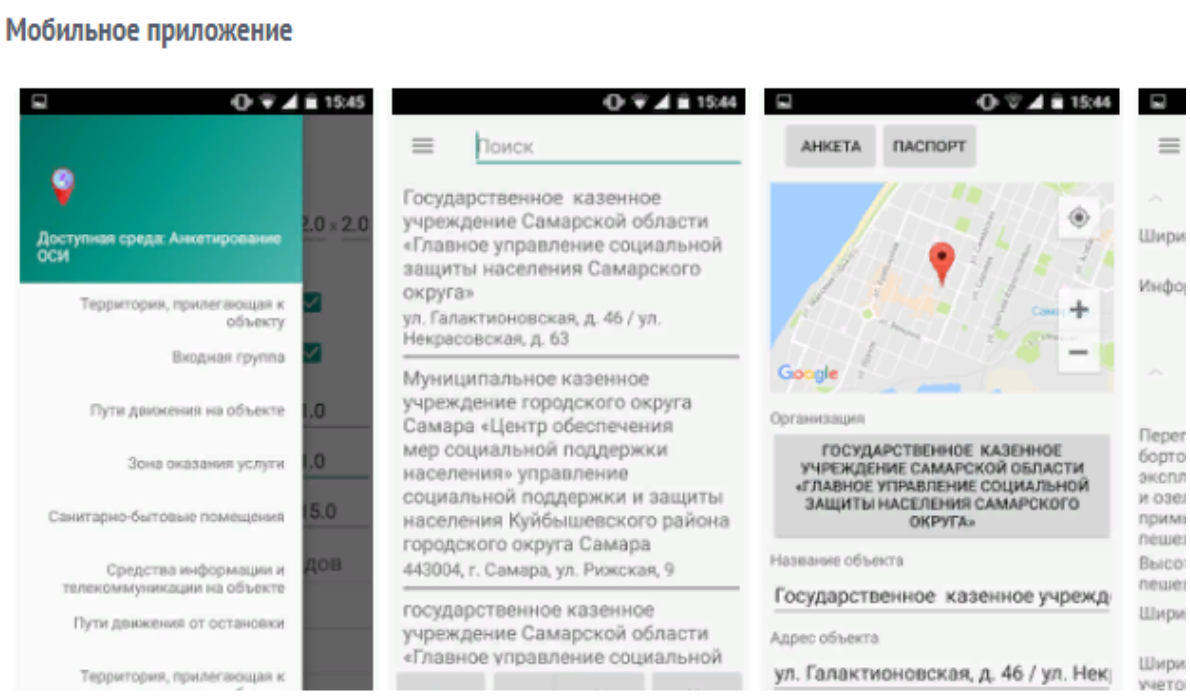


Рисунок 7 – Мобильное приложение ГИС «Доступная среда»

### 1.2.2 ДубльГис

ДубльГИС [10] Нижнего Новгорода создал дополнительный слой «Доступная архитектура для маломобильных групп».

Он содержит информацию о светофорах со звуком, улицах и доступных зданиях для маломобильных горожан - людей с ограниченными возможностями, пожилых людей, матерей с колясками и маленьких детей. Общественная организация пользователей инвалидных колясок «Инватур» совместно с местным отделением ДубльГис доработали приложение, добавив больше объектов (в настоящее время двести девяносто три организации), информацию о светофорах со звуковым уведомлением (сейчас их тридцать четыре) и увеличен список доступных улиц.

Сегодня дополнительный слой содержит триста шестьдесят шесть объектов. Все они отмечены на карте Нижнего Новгорода специальными значками: доступные здания, светофоры, улицы. Объекты сопровождаются фотографиями, информацией о наличии пандусов или перил, ссылками на подробные маршруты движения по городу.

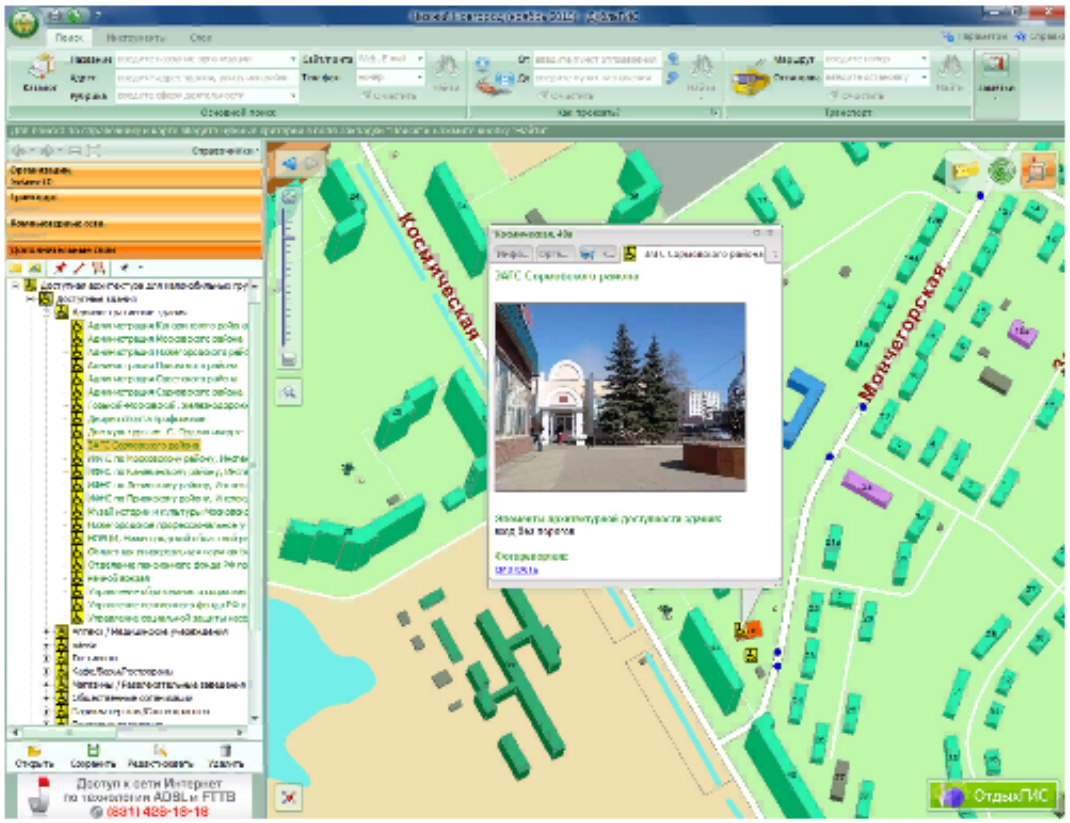


Рисунок 8 – Интерфейс дополнительного слоя «ДубльГИС»

### 1.2.3 Woof

WOOOF [11] нельзя назвать прямым аналогом разрабатываемой системы - это мобильное приложение для собаководов. Однако, несмотря на различия в целевой аудитории, этот аналог является наиболее близким по функционалу.

WOOOF содержит в себе «живую» карту, трекер прогулок и социальную сеть. Постоянно обновляемая карта позволяет получить разнообразную информацию о местонахождении ближайших зоомагазинов, ветеринарных клиник, собачьих площадок, сведения о друзьях, гуляющих поблизости, недавнее место встречи «догхантеров», и так далее. Для того, чтобы добавить предупреждение об опасности или другие полезные сведения необходимо зарегистрироваться в приложении, создав профиль себя и своего любимца. Места для прогулок можно оценивать, делиться ими с друзьями на других ресурсах, а также добавить в список друзей тех, кто может составить компанию на прогулке.


Рисунок 9 – Карта «Woof».


Рисунок 10 – Трекер прогулок «Woof»

Также приложение содержит специальный модуль «Трекер», который помогает следить за длительностью, дистанцией прогулок и активностью

питомца во время выгула. Это может представлять интерес как для владельцев крупных собак, которые нуждаются в постоянной физической активности, так и для собаководов-спортсменов, чтобы узнать, какое расстояние от дома они преодолели. Данные сохраняются и анализируются, формируя статистику для сравнения по дням, неделям и месяцам.

### 1.2.4 Портал государственной программы «Доступная среда»

Портал государственной программы «Доступная среда» [12] имеет раздел «Карта доступности объектов». Здесь отображена информация о количестве доступных объектов в каждом субъекте Российской Федерации. Так же объекты отсортированы по признакам доступности, в соответствии с категориями инвалидности и типом учреждения, например, «Здравоохранение», «Образование» или «Жилые помещения». В соответствии с выбранными параметрами на карте отображаются только удовлетворяющие им места. Так же можно указать адрес или название учреждения, для получения сведений о его доступности. Зарегистрироваться на портале могут только региональные операторы. Пользователям остаётся лишь просматривать выложенные там материалы и следить за их обновлениями.

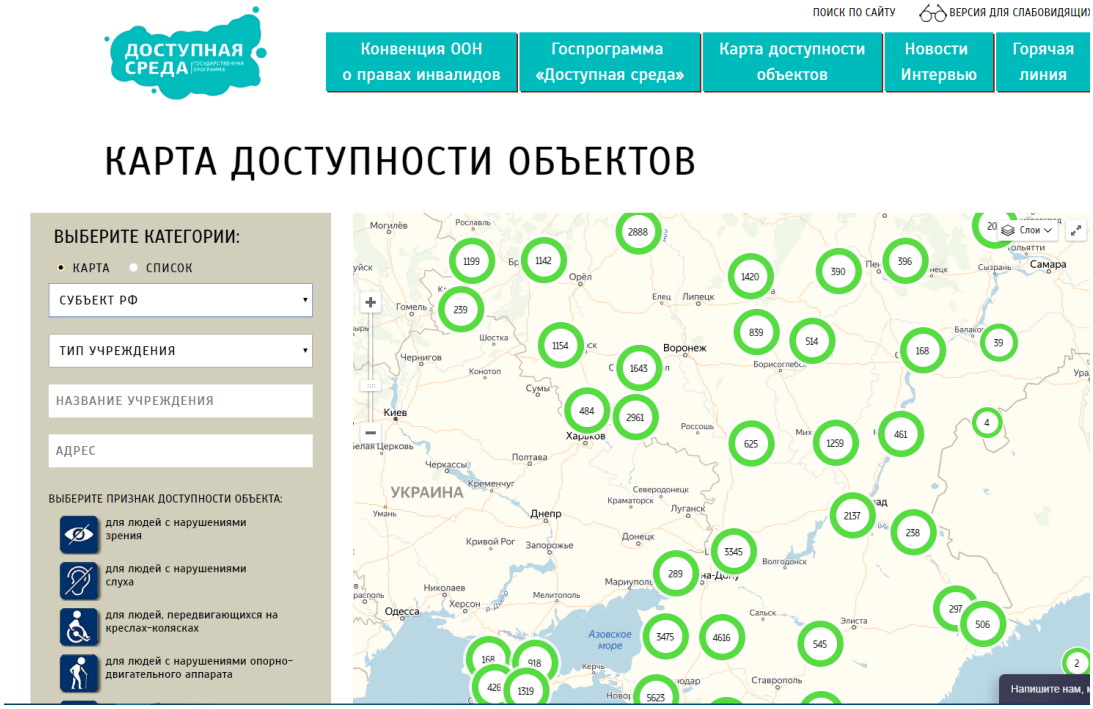

Рисунок 11 – Карта доступности объектов портала «Доступная среда»

### 1.2.5 Социальный навигатор

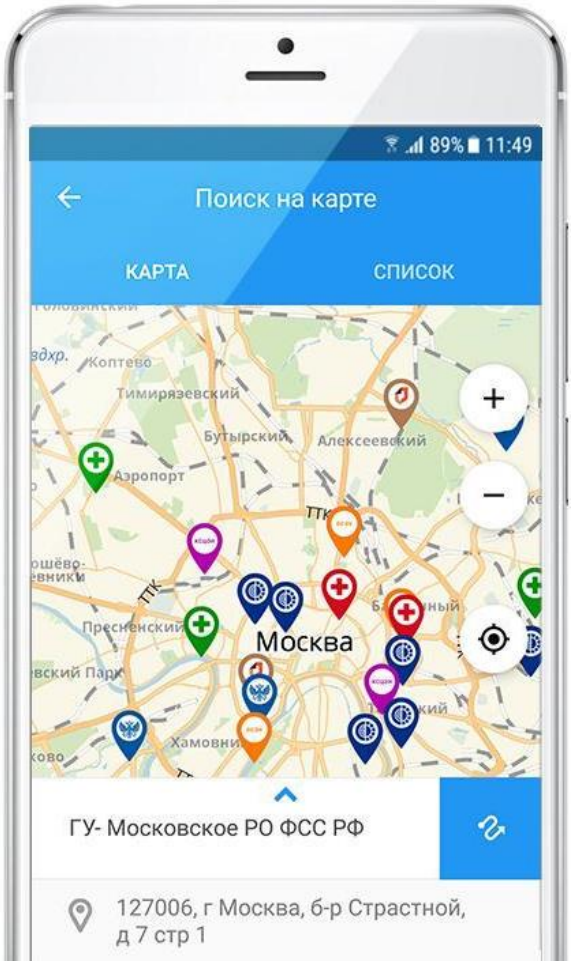


Рисунок 12 – Поиск на карте в приложении «Социальный навигатор»

«Социальный навигатор» [13] - мобильное приложение Фонда Социального Страхования Российской Федерации. В его основные функции входит поиск на карте социально значимых объектов: фондов, больниц, аптек и т.д. Эта опция позволяет найти ближайший объект, узнать режим его работы. Также есть возможность построить оптимальный маршрут из любой точки города для быстрого беспрепятственного доступа к точке назначения. Сервис доступен для операционной системы Android. Авторизация происходит через учётную запись Госуслуг. Поиск оптимального маршрута производится с использованием средств сервиса Google Maps.

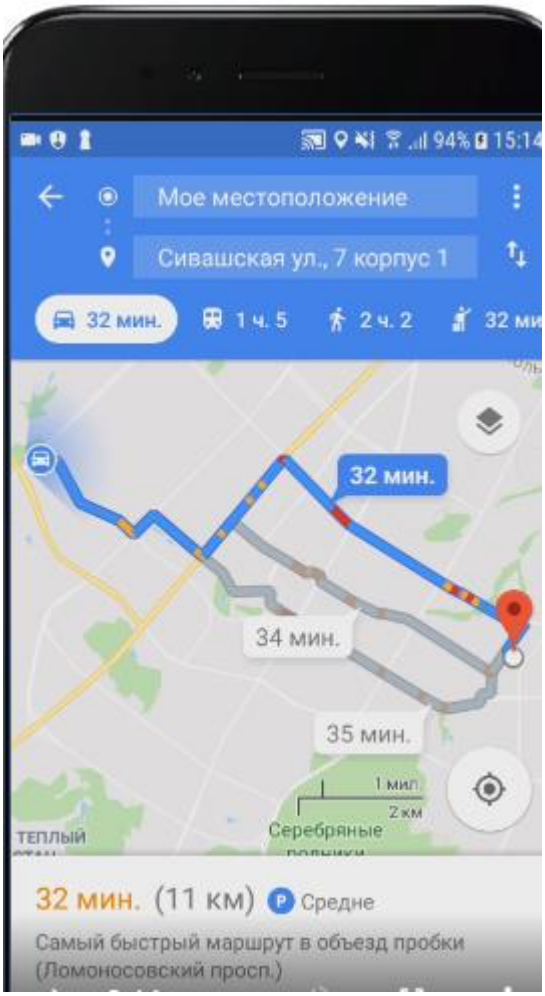


Рисунок 13 – Построение оптимального маршрута в приложении «Социальный навигатор»

### 1.3 Сравнительный анализ существующих решений

На основе анализа существующих решений геоинформационных систем для маломобильных групп населения были определены ключевые факторы: целевая группа, платформа, возможность регистрации и авторизации, возможность добавления пользовательских меток на карту, возможность комментирования меток, возможность оценивания меток, формирование усредненного рейтинга доступности районов.

На основе приведенных факторов был проведен сравнительный анализ. Результаты анализа представлены в табл. 1.

Таблица 1 - Сравнительный анализ существующих решений

| | ГИС «Доступная среда» | ДубльГИС | WOOF | Портал «Доступная среда» | Социальный навигатор |
|---|---|---|---|---|---|
| Целевая группа | маломобильные граждане | маломобильные граждане | собаководы | маломобильные граждане | маломобильные граждане |
| Платформа | мобильная/веб | мобильная/веб | мобильная | веб | мобильная |
| Регистрация пользователя | - | - | + | - | - |
| Добавление собственных меток на карту | - | - | + | - | - |
| Комментирование меток | - | - | + | - | - |
| Оценивание меток | - | - | + | + | - |
| Рейтинг доступности районов города | - | + | - | - | - |

На основе результатов сравнительного анализа существующих решений было принято решение о разработке программного обеспечения геоинформационной системы для маломобильных групп населения «Путеводная нить». Такое решение принято на основании того, что ни одно

из существующих решений не обладает полным набором требуемых параметров.

Исходя из поставленной цели работы, был определен перечень ключевых функциональных возможностей разрабатываемого программного обеспечения:

- Добавление, удаление, редактирование меток (маркеров) на карте;
- Элементы социальной сети (фотографии, оценки, комментарии);
- Модерация меток и комментариев;
- Регистрация и авторизация пользователей.

# 2 ОПИСАНИЕ РАЗРАБАТЫВАЕМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## 2.1 Этапы разработки геоинформационной системы для маломобильных групп населения

Для создания программного обеспечения с требуемым набором функциональных возможностей необходимо пройти каждый этап проектирования и разработки геоинформационной системы:

- Выбор языков программирования;
- Выбор платформ разработки;
- Выбор СУБД;
- Проектирование и разработка базы данных;
- Разработка серверной части программного обеспечения;
- Разработка клиентской части программного обеспечения;
- Проверка работоспособности разработанной системы.

## 2.2 Функциональные возможности геоинформационной системы

Система позволяет маркировать точки на карте города и, в соответствии с доступностью для маломобильных граждан, оценивать доступность по пятибалльной шкале от 0 до 5 баллов, оставлять комментарии. На основе оценок и отзывов формируется рейтинг районов по доступности среды, появляется возможность выбрать наиболее комфортный для перемещения район города. Предусмотрено три роли доступа: «Пользователь», «Администратор» и «Модератор». При регистрации пользователю по умолчанию присваивается стандартная роль «Пользователь».

Функциональные возможности роли «Пользователь»:

- Добавление меток на карту;
- Добавление оценки метки;
- Добавление комментария;

- Добавление фотографии метки;
- Просмотр карты и рейтинга доступности.

Роль «Модератор» может быть присвоена пользователю только другим пользователем с ролью «Администратор». Функциональные возможности роли «Модератор» включают в себя все возможности роли «Пользователь», а также следующие возможности:

- Модерация меток;
- Модерация комментариев;
- Модерация фотографий.

Роль «Администратор» может быть присвоена пользователю только другим пользователем с ролью «Администратор». Функциональные возможности роли «Администратор» включают в себя все возможности роли «Модератор», а также следующие возможности:

- Активация и деактивация учетных записей;
- Изменение ролей пользователей.

Для наиболее наглядной демонстрации функционала системы была построена диаграмма прецедентов (вариантов использования) (рис. 14).

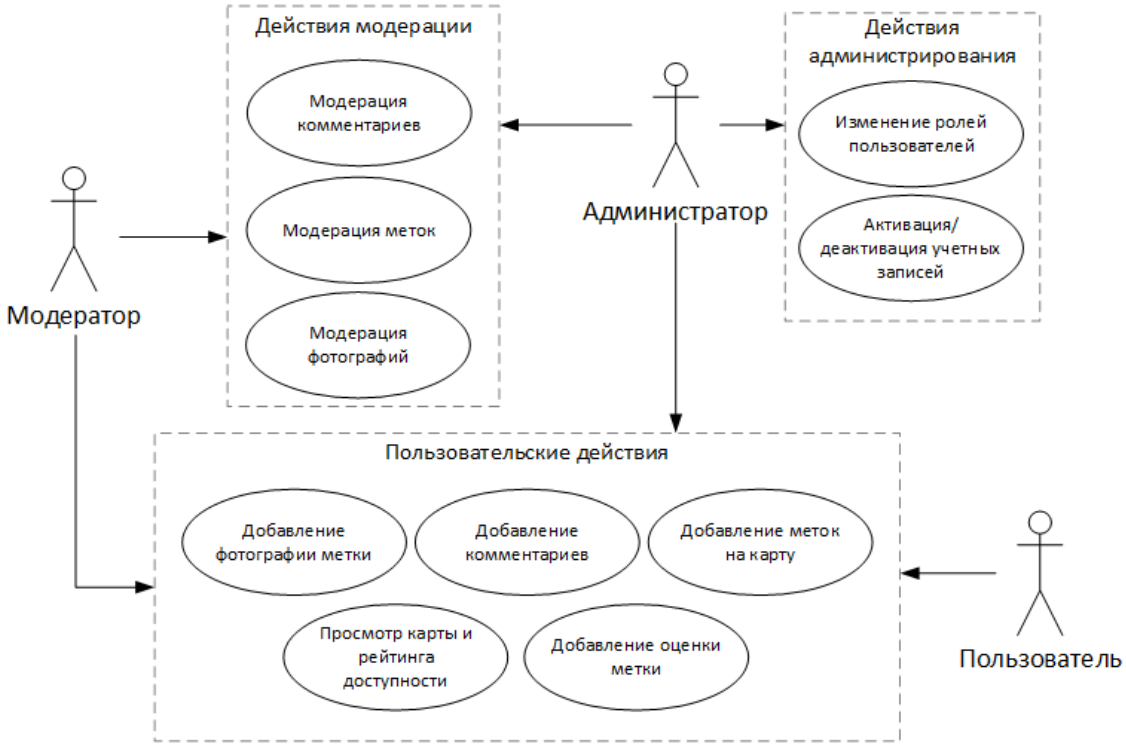

Рисунок 14 – Диаграмма вариантов использования ГИС «Путеводная нить»

## 2.3 Входные и выходные данные

### 2.3.1 Регистрация пользователя

Входные данные (вводятся незарегистрированным пользователем):

- Фамилия;
- Имя;
- Email;
- Логин;
- Пароль;
- Подтверждение пароля;
- Пол.

Выходные данные:

- Электронное письмо с информацией о регистрации (отправляется на указанный при регистрации email);
- Элемент в списке пользователей в администраторской панели (доступно только пользователю с ролью «Администратор»). Элемент содержит введенные при регистрации данные, роль «Пользователь» и является деактивированным по умолчанию.

### 2.3.2 Авторизация пользователя

Входные данные (вводятся неавторизованным пользователем):

- Логин;
- Пароль.

Выходные данные:

- Ссылки главного меню, в зависимости от роли пользователя.

### 2.3.3 Просмотр карты с данными

Входные данные:

- Границы области просмотра карты (прямоугольник);
- Список искомых статусов маркеров (необязательно);

- Список искомых статусов комментариев маркеров (необязательно);
- Искомый текст (необязательно);
- Флаг отображения только маркеров текущего пользователя (необязательно);
- Флаг отображения сетки.

Выходные данные:

- Карта в рамках выбранной области просмотра;
- Найденные маркеры;
- Сетка с цветовой индикацией рейтинга доступности (если подан на вход флаг отображения сетки).

### 2.3.4 Добавление маркера

Входные данные:

- Координаты (широта и долгота);
- Оценка доступности;
- Описание.

Выходные данные:

- Маркер на карте (доступен только пользователям с ролями «Администратор» или «Модератор»).

### 2.3.5 Добавление комментария

Входные данные:

- Идентификатор маркера;
- Текст комментария.

Выходные данные:

- Комментарий (текст, дата и время добавления, логин автора и статус «На модерации») в списке комментариев выбранного маркера. Доступно только пользователям с ролями «Администратор» или «Модератор».

### 2.3.6 Добавление фотографии

Входные данные:

- Идентификатор маркера;

- Файл с фотографией;
- Описание.

Выходные данные:

- Фотография (изображение и описание) в списке фотографий выбранного при добавлении маркера.

### 2.3.7 Выставление оценок

Входные данные:

- Идентификатор маркера;
- Оценка от 0 до 5.

Выходные данные:

- Усредненная оценка маркера с учетом поставленной.

## 2.4 Структура разрабатываемого программного обеспечения

Для обеспечения всех описанных ранее функциональных возможностей и разграничения прав пользователей разрабатываемое программное обеспечение должно быть реализовано в виде трехуровневой клиент-серверной системы, состоящей из следующих уровней:

1. База данных, управляемая сервером СУБД;
2. Серверная часть, выполняемая сервером приложений;
3. Клиентская часть, выполняемая в браузере пользователя.

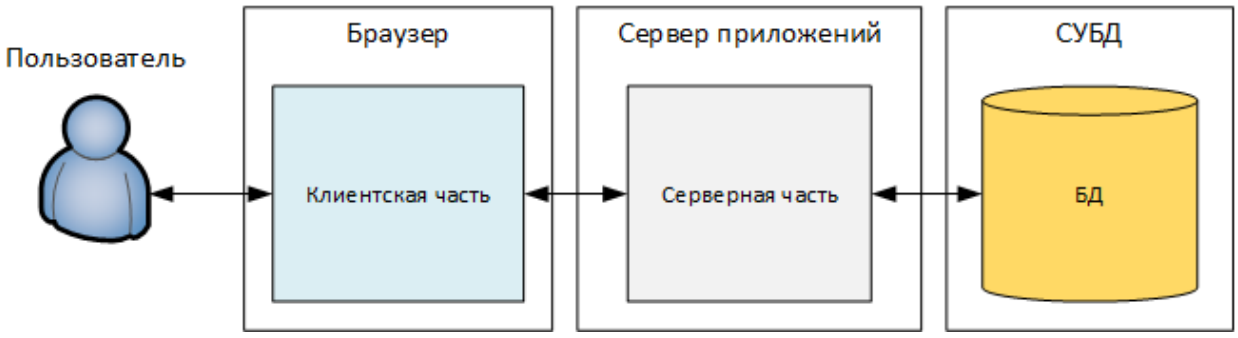


Рисунок 15 – Структурная схема системы

Такая архитектура позволит поддерживать одновременную работу многих пользователей с разнородных устройств.

## 2.5 Структура базы данных

Проектирование базы данных является одной из самых важных этапов в разработке программного обеспечения. На основе сущностей в БД и связей между ними строится модель данных всего программного обеспечения.

Для работы геоинформационной системы для маломобильных групп населения «Путеводная нить» необходимы следующие сущности БД: Пользователи, Аватары пользователей, Фотографии, Метки, Оценки, Комментарии, Статусы, Настройки.

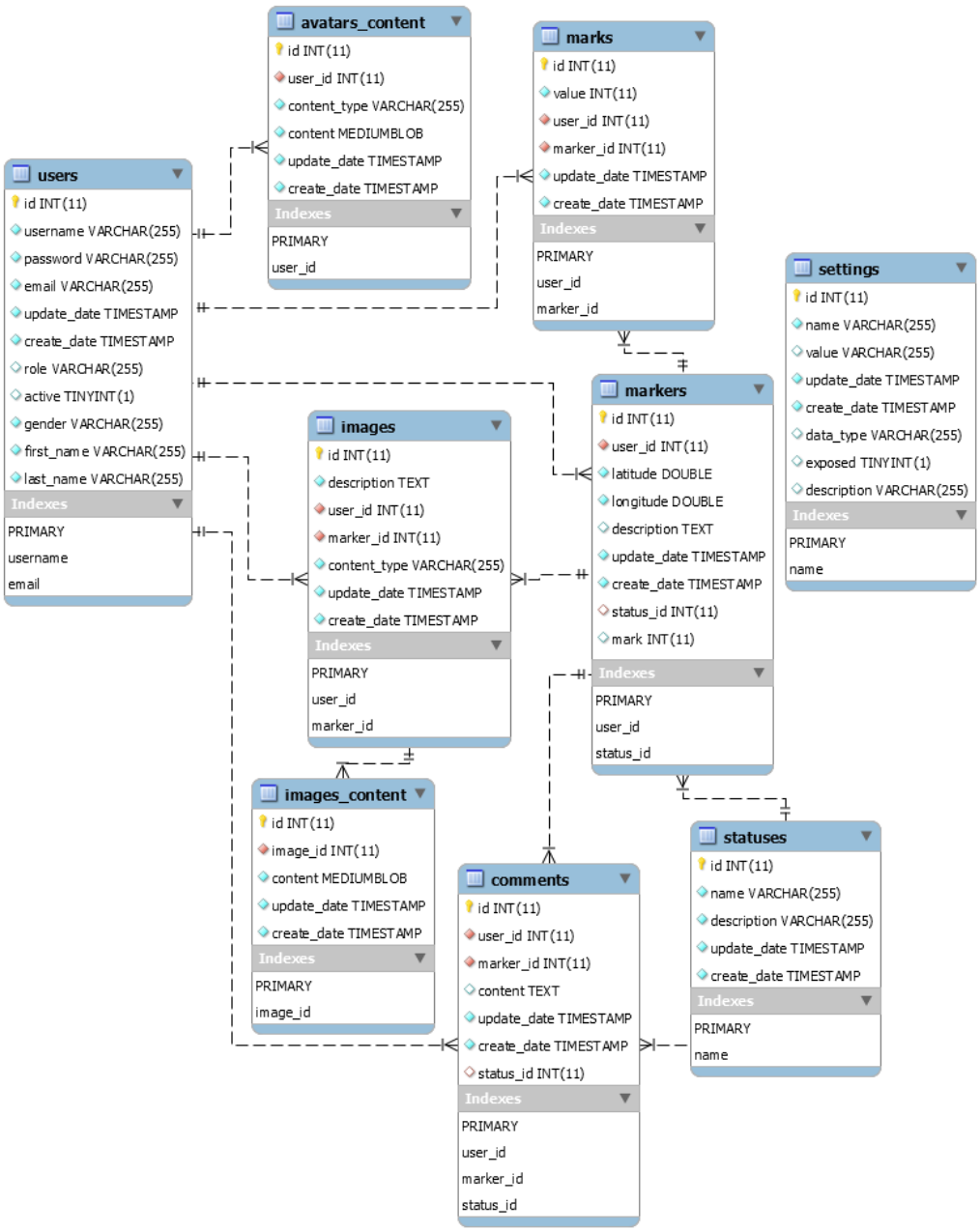

Рисунок 16 – Схема базы данных

Приведем описание таблиц базы данных.

Таблица 2 - Таблица «users»

| Имя | Тип | Атрибуты |
|---|---|---|
| id | Int (11) | Первичный ключ |
| username | Varchar (255) | Уникальный индекс |
| password | Varchar (255) | |
| email | Varchar (255) | Уникальный индекс |
| update_date | Timestamp | |
| create_date | Timestamp | |
| role | Varchar (255) | |
| active | TinyInt (1) | |
| gender | Varchar (255) | |
| first_name | Varchar (255) | |
| last_name | Varchar (255) | |

Таблица “users” предназначена для хранения данных пользователей геоинформационной системы. Таблица имеет следующие столбцы: id – идентификатор пользователя, username – логин, password – пароль, email – адрес электронной почты, update_date – дата обновления профиля, create_date – дата создания профиля, role – роль, active – отметка модератора об активации учетной записи, gender – пол, first_name – имя, last_name – фамилия.

Таблица 3 - Таблица «avatars_content»

| Имя | Тип | Атрибуты |
|---|---|---|
| id | Int (11) | Первичный ключ |
| user_id | Int (11) | Внешний ключ, индекс |
| content_type | Varchar (255) | |
| content | Mediumblob | |
| update_date | Timestamp | |
| create_date | Timestamp | |

Таблица “avatars_content” предназначена для хранения данных об аватарах пользователей. Таблица имеет следующие столбцы: id – идентификатор, user_id – идентификатор пользователя (внешний ключ к таблице users), content_type – тип хранимого содержимого, content – содержимое, update_date – дата обновления аватара, create_date – дата создания аватара.

Таблица 4 - Таблица «images»

| Имя | Тип | Атрибуты |
|---|---|---|
| id | Int (11) | Первичный ключ |
| description | Text | |
| user_id | Int (11) | Внешний ключ, индекс |
| marker_id | Int (11) | Внешний ключ, индекс |
| content_type | Varchar (255) | |
| update_date | Timestamp | |
| create_date | Timestamp | |

Таблица “images” предназначена для хранения данных о фото к меткам. Таблица имеет следующие столбцы: id – идентификатор, description – описание, user_id – идентификатор пользователя (внешний ключ к таблице users), marker_id – идентификатор метки (внешний ключ к таблице markers), content_type – тип хранимого содержимого, update_date – дата обновления фотографии, create_date – дата создания фотографии.

Таблица 5 - Таблица «images_content»

| Имя | Тип | Атрибуты |
|---|---|---|
| id | Int (11) | Первичный ключ |
| image_id | Int (11) | Внешний ключ, индекс |
| content | Mediumblob | |
| update_date | Timestamp | |
| create_date | Timestamp | |

Таблица “images_content” предназначена для хранения непосредственно фото к меткам. Таблица имеет следующие столбцы: id – идентификатор, image_id – идентификатор картинки (внешний ключ к таблице images), content – содержимое, update_date – дата обновления содержимого, create_date – дата создания.

Таблица 6 - Таблица «markers»

| Имя | Тип | Атрибуты |
|---|---|---|
| id | Int (11) | Первичный ключ |
| user_id | Int (11 | Внешний ключ, индекс |
| latitude | Double | |
| longitude | Double | |
| description | Text | |
| update_date | Timestamp | |
| create_date | Timestamp | |
| status_id | Int (11) | Внешний ключ, индекс |
| mark | Int (11) | |

Таблица “markers” предназначена для хранения данных о метках. Таблица имеет следующие столбцы: id – идентификатор, user_id – идентификатор пользователя (внешний ключ к таблице users), latitude – широта, longitude – долгота, update_date – дата обновления метки, create_date – дата создания метки, status_id – идентификатор статуса метки (внешний ключ к таблице status), mark – оценка автора.

Таблица 7 - Таблица «marks»

| Имя | Тип | Атрибуты |
|---|---|---|
| id | Int (11) | Первичный ключ |
| value | Int (11) | |
| user_id | Int (11) | Внешний ключ, индекс |
| marker_id | Int (11) | Внешний ключ, индекс |

| | | |
|---|---|---|
| update_date | Timestamp | |
| create_date | Timestamp | |

Таблица “marks” предназначена для хранения данных об оценках. Таблица имеет следующие столбцы: id – идентификатор, value – значение, user_id – идентификатор пользователя (внешний ключ к таблице users), marker_id – идентификатор метки (внешний ключ к таблице markers), update_date – дата обновления оценки, create_date – дата создания оценки.

Таблица 8 - Таблица «comments»

| Имя | Тип | Атрибуты |
|---|---|---|
| id | Int (11) | Первичный ключ |
| user_id | Int (11) | Внешний ключ, индекс |
| marker_id | Int (11) | Внешний ключ, индекс |
| content | Text | |
| update_date | Timestamp | |
| create_date | Timestamp | |
| status_id | Int (11) FK | Внешний ключ, индекс |

Таблица “comments” предназначена для хранения данных о комментариях. Таблица имеет следующие столбцы: id – идентификатор, user_id – идентификатор пользователя (внешний ключ к таблице users), marker_id – идентификатор метки (внешний ключ к таблице markers), content-контент, update_date – дата обновления комментария, create_date – дата создания комментария, status_id – идентификатор статуса метки (внешний ключ к таблице statuses).

Таблица 9 - Таблица «statuses»

| Имя | Тип | Атрибуты |
|---|---|---|
| id | Int (11) | Первичный ключ |
| name | Varchar (255) | Уникальный индекс |

| | | |
|---|---|---|
| update_date | Timestamp | |
| create_date | Timestamp | |
| description | Varchar (255) | |

Таблица “statuses” предназначена для хранения данных о статусах. Таблица имеет следующие столбцы: id – идентификатор, name – название статуса, update_date – дата обновления статуса, create_date – дата создания статуса, description – описание статуса.

Таблица 10 - Сущность «settings»

| Имя | Тип | Атрибуты |
|---|---|---|
| id | Int (11) | Первичный ключ |
| name | Varchar (255) | Уникальный индекс |
| value | Varchar (255) | |
| update_date | Timestamp | |
| create_date | Timestamp | |
| data_type | Varchar (255) | |
| exposed | TinyInt (1) | |

Таблица “settings” предназначена для хранения данных о настройках приложения. Таблица имеет следующие столбцы: id – идентификатор, name – название настройки, update_date – дата обновления настройки, create_date – дата создания настройки, data_type – тип данных, exposed – флаг доступности настройки для изменения.

### 2.6 Структура серверной части

Серверная часть приложения должна реализовывать следующие функции:

- Взаимодействие с БД;
- Обработка HTTP-запросов от клиентской части и формирование HTTP-ответов;
- Бизнес-логика приложения;
- Разграничение доступа к операциям.

Для реализации данных функций было принято решение построить серверную часть на основе многослойной архитектуры (рис. 17).

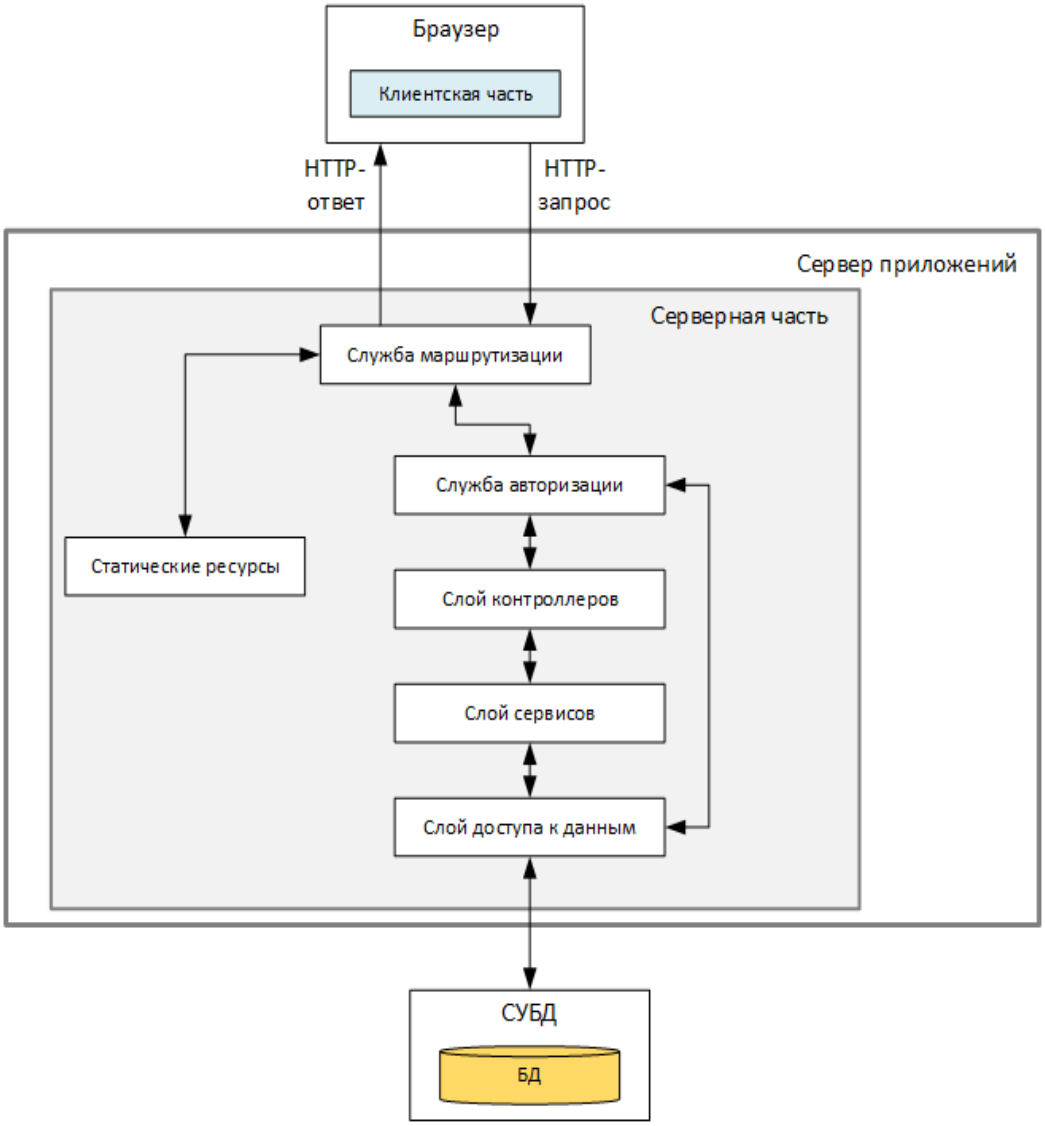


Рисунок 17 – Структурная схема серверной части

В структуре серверной части выделены следующие компоненты:

- Слой доступа к данным предназначен для соединения с БД, формирования и выполнения SQL-запросов, преобразования результатов запросов к нужному приложению формату;
- Слой сервисов предназначен для реализации основной бизнес-логики;
- Слой контроллеров предназначен для обработки данных, полученных из HTTP-запросов и вызова нужных действия слоя сервисов;

- Служба авторизации определяет ограничения доступа к действияем контроллеров;
- Служба маршрутизации в зависимости от данных HTTP-запроса определяет, какое действие контроллеров должно быть вызвано или какой статический ресурс возвращен;
- Статические ресурсы включают клиентские скрипты, стили и изображения.

### 2.7 Взаимодействие серверной и клиентской части

Взаимодействие клиентской и серверной части должно осуществляться по протоколу HTTP с помощью веб-сервисов. Приведем список необходимых методов веб-сервисов серверной части.

#### 2.7.1 Аутентификация пользователя

Запрос:

POST /api/auth/authenticate

Тип: JSON

Пример тела запроса:

```
{
  "body": {
    "password": "string",
    "username": "string"
  }
}
```

Ответ:

Тип: JSON

Пример тела ответа:

```
{
  "body": {
    "active": true,
    "createDate": "2020-05-21T08:59:48.554Z",
    "email": "string",
    "firstName": "string",
    "gender": {
      "description": "string",
```

```
      "name": "string"
    },
    "id": 0,
    "lastName": "string",
    "role": {
      "description": "string",
      "name": "string"
    },
    "updateDate": "2020-05-21T08:59:48.554Z",
    "username": "string"
  }
}
```


### 2.7.2 Выход из учетной записи

Запрос: POST /api/auth/logout

Без тела.

Ответ:

Тип: JSON

Пример тела ответа:


```
{
  "body": {
    "value": true
  }
}
```


### 2.7.3 Получение информации о текущем пользователе

Запрос: GET /api/auth/my

Ответ:

Тип: JSON

Пример тела ответа:


```
{
  "body": {
    "active": true,
    "createDate": "2020-05-21T09:09:32.646Z",
    "email": "string",
    "firstName": "string",
    "gender": {
      "description": "string",
```

```
      "name": "string"
    },
    "id": 0,
    "lastName": "string",
    "role": {
      "description": "string",
      "name": "string"
    },
    "updateDate": "2020-05-21T09:09:32.646Z",
    "username": "string"
  }
}
```


### 2.7.4 Обновление пароля

Запрос: POST /api/auth/updatePassword

Тип: JSON

Пример тела запроса:


```
{
  "body": {
    "confirmation": "string",
    "newPassword": "string",
    "oldPassword": "string"
  }
}
```


Ответ:

Тип: JSON

Пример тела ответа:


```
{
  "body": {
    "active": true,
    "createDate": "2020-05-21T09:11:41.129Z",
    "email": "string",
    "firstName": "string",
    "gender": {
      "description": "string",
      "name": "string"
    },
    "id": 0,
    "lastName": "string",
    "role": {
      "description": "string",
```

```
      "name": "string"
    },
    "updateDate": "2020-05-21T09:11:41.129Z",
    "username": "string"
  }
}
```

### 2.7.5 Регистрация пользователя

Запрос: POST /api/auth/register

Тип: JSON

Пример тела запроса:

```
{
  "body": {
    "confirmation": "string",
    "email": "string",
    "firstName": "string",
    "gender": "MALE",
    "lastName": "string",
    "password": "string",
    "username": "string"
  }
}
```

Ответ:

Тип: JSON

Пример тела ответа:

```
{
  "body": {
    "active": true,
    "createDate": "2020-05-21T09:12:35.343Z",
    "email": "string",
    "firstName": "string",
    "gender": {
      "description": "string",
      "name": "string"
    },
    "id": 0,
    "lastName": "string",
    "role": {
      "description": "string",
      "name": "string"
    },
```

```
    "updateDate": "2020-05-21T09:12:35.343Z",
    "username": "string"
  }
}
```


### 2.7.6 Получение аватара пользователя

Ответ: GET /api/avatar/{id}

Тип: изображение

### 2.7.7 Регистрация пользователя

Запрос: POST /api/avatar/delete

Без тела.

Ответ:

Тип: JSON

Пример тела ответа:


```
{
  "body": {
    "value": true
  }
}
```


### 2.7.8 Загрузка аватара

Запрос: POST /api/avatar/upload

Тип: файл.

Ответ:

Тип: JSON

Пример тела ответа:


```
{
  "body": {
    "value": 0
  }
}
```


### 2.7.9 Одобрение комментария

Запрос: POST /api/comment/{id}/approve

Без тела.

Ответ:

Тип: JSON

Пример тела ответа:


```
{
  "body": {
    "content": "string",
    "createDate": "2020-05-21T09:18:24.587Z",
    "id": 0,
    "status": {
      "description": "string",
      "name": "string"
    },
    "user": {
      "id": 0,
      "username": "string"
    }
  }
}
```


### 2.7.10 Отклонение комментария

Запрос: POST /api/comment/{id}/decline

Без тела.

Ответ:

Тип: JSON

Пример тела ответа:


```
{
  "body": {
    "content": "string",
    "createDate": "2020-05-21T09:18:24.587Z",
    "id": 0,
    "status": {
      "description": "string",
      "name": "string"
    },
    "user": {
      "id": 0,
```

```
      "username": "string"
    }
  }
}
```

### 2.7.11 Получения изображения

Запрос: GET /api/image/{id}

Ответ:

Тип: изображение.

### 2.7.12 Удаление изображения

Запрос: POST /api/image/{id}/delete

Без тела.

Ответ:

Тип: JSON

Пример тела ответа:

```
{
  "body": {
    "value": true
  }
}
```

### 2.7.13 Получения объектов карты

Запрос: POST /api/map/search

Тип: JSON

Пример тела:

```
{
  "body": {
    "calculateRectangles": true,
    "commentStatuses": [
      "ON_APPROVAL"
    ],
    "latBottomLeft": 0,
    "latTopRight": 0,
    "lngBottomLeft": 0,
    "lngTopRight": 0,
```

```
    "onlyMy": true,
    "searchText": "string",
    "statuses": [
      "ON_APPROVAL"
    ]
  }
}
```

Ответ:

Тип: JSON

Пример тела ответа:

```
{
  "body": {
    "markers": [
      {
        "averageMark": 0,
        "id": 0,
        "latitude": 0,
        "longitude": 0,
        "status": {
          "description": "string",
          "name": "string"
        }
      }
    ],
    "rectangles": [
      {
        "count": 0,
        "latBottomLeft": 0,
        "latTopRight": 0,
        "lngBottomLeft": 0,
        "lngTopRight": 0,
        "value": 0
      }
    ]
  }
}
```

### 2.7.14 Получения информации о маркере

Запрос: GET /api/marker/{id}

Ответ:

Тип: JSON

Пример тела ответа:


```
{
  "body": {
    "authorMark": 0,
    "averageMark": 0,
    "comments": [
      {
        "content": "string",
        "createDate": "2020-05-21T09:23:59.576Z",
        "id": 0,
        "status": {
          "description": "string",
          "name": "string"
        },
        "user": {
          "id": 0,
          "username": "string"
        }
      }
    ],
    "createDate": "2020-05-21T09:23:59.576Z",
    "description": "string",
    "id": 0,
    "images": [
      {
        "createDate": "2020-05-21T09:23:59.576Z",
        "description": "string",
        "id": 0,
        "user": {
          "id": 0,
          "username": "string"
        }
      }
    ],
    "latitude": 0,
    "longitude": 0,
    "myMark": 0,
    "status": {
      "description": "string",
      "name": "string"
    },
    "user": {
      "id": 0,
```

```
      "username": "string"
    }
  }
}
```

### 2.7.15 Одобрение маркера

Запрос: POST /api/marker/{id}/approve

Без тела.

Ответ:

Тип: JSON

Пример тела ответа:

```
{
  "body": {
    "averageMark": 0,
    "id": 0,
    "latitude": 0,
    "longitude": 0,
    "status": {
      "description": "string",
      "name": "string"
    }
  }
}
```

### 2.7.16 Отклонение маркера

Запрос: POST /api/marker/{id}/decline

Без тела.

Ответ:

Тип: JSON

Пример тела ответа:

```
{
  "body": {
    "averageMark": 0,
    "id": 0,
    "latitude": 0,
    "longitude": 0,
    "status": {
```

```
      "description": "string",
      "name": "string"
    }
  }
}
```

### 2.7.17 Удаление маркера

Запрос: POST /api/marker/{id}/delete

Без тела.

Ответ:

Тип: JSON

Пример тела ответа:

```
{
  "body": {
    "value": true
  }
}
```

### 2.7.18 Добавление комментария

Запрос: POST /api/marker/{id}/comment/add

Тип тела: JSON

Пример тела:

```
{
  "body": {
    "content": "string"
  }
}
```

Ответ:

Тип: JSON

Пример тела ответа:

```
{
  "body": {
    "content": "string",
    "createDate": "2020-05-21T09:29:01.064Z",
    "id": 0,
    "status": {
      "description": "string",
```

```
      "name": "string"
    },
    "user": {
      "id": 0,
      "username": "string"
    }
  }
}
```

### 2.7.19 Загрузка изображения

Запрос: POST /api/marker/{id}/image/upload

Тип: файл.

Ответ:

Тип: JSON

Пример тела ответа:

```
{
  "body": {
    "createDate": "2020-05-21T09:30:11.201Z",
    "description": "string",
    "id": 0,
    "user": {
      "id": 0,
      "username": "string"
    }
  }
}
```

### 2.7.20 Выставление оценки

Запрос: POST /api/marker/{id}/mark

Тип тела: JSON

Пример тела:

```
{
  "body": {
    "value": 0
  }
}
```

Ответ:

Тип: JSON

Пример тела ответа:


```
{
  "body": {
    "authorMark": 0,
    "averageMark": 0,
    "comments": [
      {
        "content": "string",
        "createDate": "2020-05-21T09:31:30.476Z",
        "id": 0,
        "status": {
          "description": "string",
          "name": "string"
        },
        "user": {
          "id": 0,
          "username": "string"
        }
      }
    ],
    "createDate": "2020-05-21T09:31:30.476Z",
    "description": "string",
    "id": 0,
    "images": [
      {
        "createDate": "2020-05-21T09:31:30.476Z",
        "description": "string",
        "id": 0,
        "user": {
          "id": 0,
          "username": "string"
        }
      }
    ],
    "latitude": 0,
    "longitude": 0,
    "myMark": 0,
    "status": {
      "description": "string",
      "name": "string"
    },
    "user": {
      "id": 0,
      "username": "string"
```

```
    }
 }
}
```

### 2.7.21 Добавление маркера

Запрос: POST /api/marker/add

Тип тела: JSON

Пример тела:

```
{
  "body": {
    "description": "string",
    "latitude": 0,
    "longitude": 0,
    "mark": 0
  }
}
```

Ответ:

Тип: JSON

Пример тела ответа:

```
{
  "body": {
    "averageMark": 0,
    "id": 0,
    "latitude": 0,
    "longitude": 0,
    "status": {
      "description": "string",
      "name": "string"
    }
  }
}
```

### 2.7.22 Обновление данных пользователя

Запрос: POST /api/user/{id}

Тип тела: JSON

Пример тела:

```
{
  "body": {
    "active": true,
```

```
    "email": "string",
    "role": "ROLE_ADMIN"
  }
}
```

Ответ:

Тип: JSON

Пример тела ответа:

```
{
  "body": {
    "active": true,
    "createDate": "2020-05-21T09:34:02.563Z",
    "email": "string",
    "firstName": "string",
    "gender": {
      "description": "string",
      "name": "string"
    },
    "id": 0,
    "lastName": "string",
    "role": {
      "description": "string",
      "name": "string"
    },
    "updateDate": "2020-05-21T09:34:02.563Z",
    "username": "string"
  }
}
```

### 2.7.23 Получения списка пользователей

Запрос: GET /api/user/all

Ответ:

Тип: JSON

Пример тела ответа:

```
{
  "body": {
    "items": [
      {
        "active": true,
```

```
        "createDate": "2020-05-21T09:36:00.962Z",
        "email": "string",
        "firstName": "string",
        "gender": {
          "description": "string",
          "name": "string"
        },
        "id": 0,
        "lastName": "string",
        "role": {
          "description": "string",
          "name": "string"
        },
        "updateDate": "2020-05-21T09:36:00.962Z",
        "username": "string"
      }
    ]
  }
}
```


### 2.7.23 Получения списка ролей

Запрос: GET /api/user/roles

Ответ:

Тип: JSON

Пример тела ответа:


```
{
  "body": {
    "items": [
      {
        "description": "string",
        "name": "string"
      }
    ]
  }
}
```


## 2.8 Структура клиентской части приложения

Серверная часть приложения должна реализовывать следующие функции:

- Взаимодействие с пользователем;
- Отправка HTTP-запросов к серверной части и обработка HTTP-ответов.

Для реализации данных функций было принято решение построить серверную часть на основе многослойной архитектуры (рис. 18).

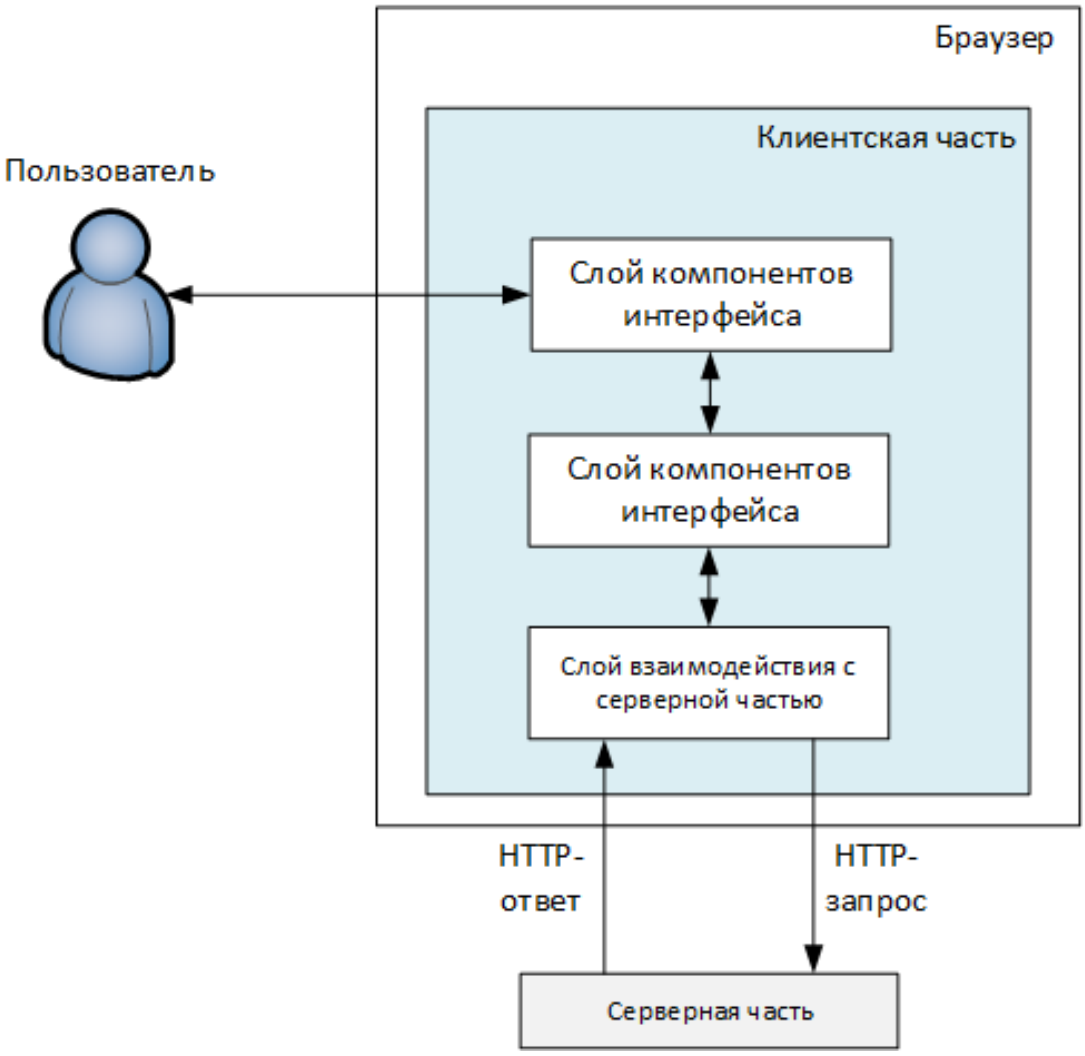


Рисунок 18 – Структурная схема клиентской части

В структуре серверной части выделены следующие компоненты:

- Слой взаимодействия с серверной частью предназначен для асинхронного обмена HTTP-сообщениями с серверной частью;
- Слой сервисов предназначен для реализации бизнес-логики;

- Слой компонентов интерфейса предназначен для отображения и динамического перестроения разметки страниц, обработки действий пользователя.

## 2.9 Инструменты разработки

В качестве СУБД для разрабатываемой системы был выбран MySQL [14].

Для разработки серверной части использовались следующие инструменты:

- Объектно-ориентированный кроссплатформенный язык программирования Java [15];
- Фреймворк Spring Framwork [16] для общей организации каркаса серверной части, а также следующие компоненты:
    - Spring Data для взаимодействия с БД;
    - Spring Security для разграничения прав доступа и авторизации;
    - Spring MVC для обработки HTTP-запросов и формирования HTTP-ответов.
- Библиотека Hibernate [17] для реализации объектно-реляционного отображения;
- Библиотека Lombok [18] для автоматической генерации типового кода;
- Библиотека Mapstruct [19] для преобразований сущностей и объектов передачи данных;
- Библиотека JJWT [20] для реализации авторизации с помощью JSON Web Token;
- Библиотека Jackson [21] для преобразований между объектами передачи данных и текстом в формате JSON;
- Библиотека Luquibase [22] для версионирования и миграций БД.

Для разработки клиентской части использовались следующие инструменты:

- Языки HTML и CSS [23];
- Мультипарадигменный язык программирования TypeScript [24, 25];
- Библиотека React для создания компонентов пользовательского интерфейса;
- Библиотека Mobx [26] для управления состоянием приложения и динамическим обновлением компонентов;
- Библиотека Axios [27] для обмена HTTP-сообщениями с серверной частью;
- Технология SASS для препроцессинга CSS [28];
- Webpack для сборки исходного кода.

# 3 ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ГЕОИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ МАЛОМОБИЛЬНЫХ ГРУПП НАСЕЛЕНИЯ «ПУТЕВОДНАЯ НИТЬ»

Для того, чтобы получить доступ к геоинформационной системе, необходимо ввести URL в адресной строке браузера.

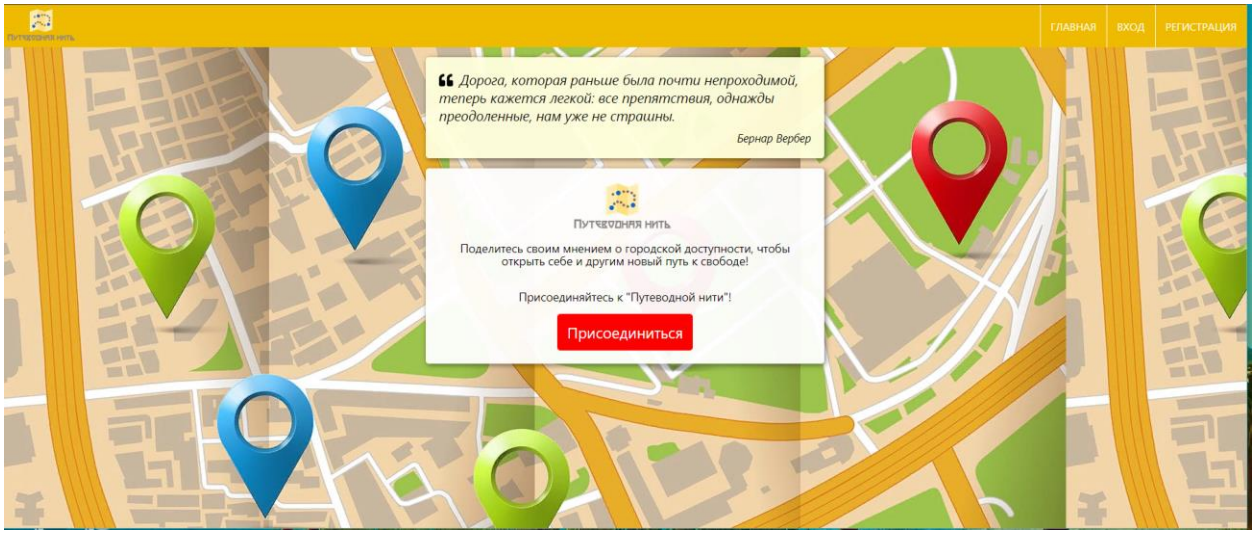


Рисунок 19 – Главная страница

Неавторизованный пользователь может зарегистрироваться в системе или авторизоваться, если уже имеет учётную запись.

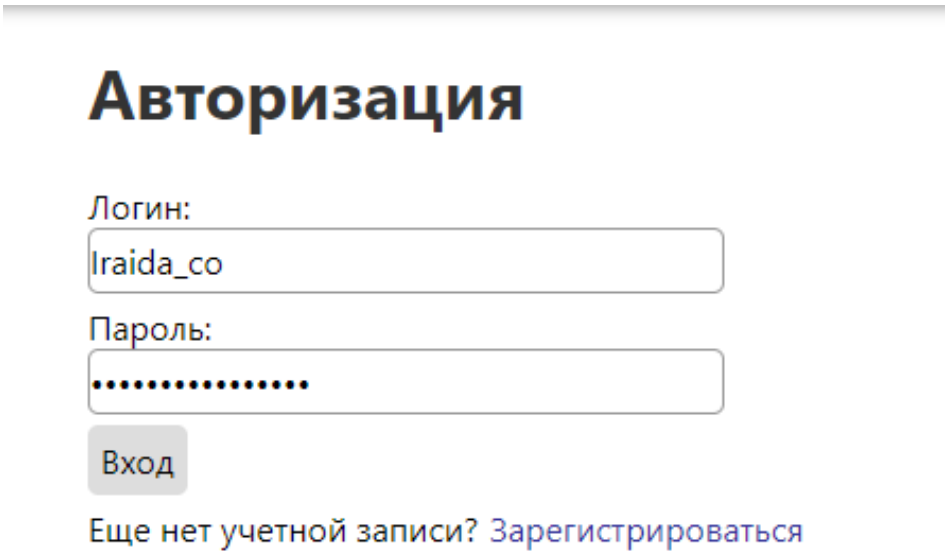


Рисунок 20 – Форма авторизации

Для создания учётной записи требуется нажать «Присоединиться» и заполнить регистрационную форму (рис.29)

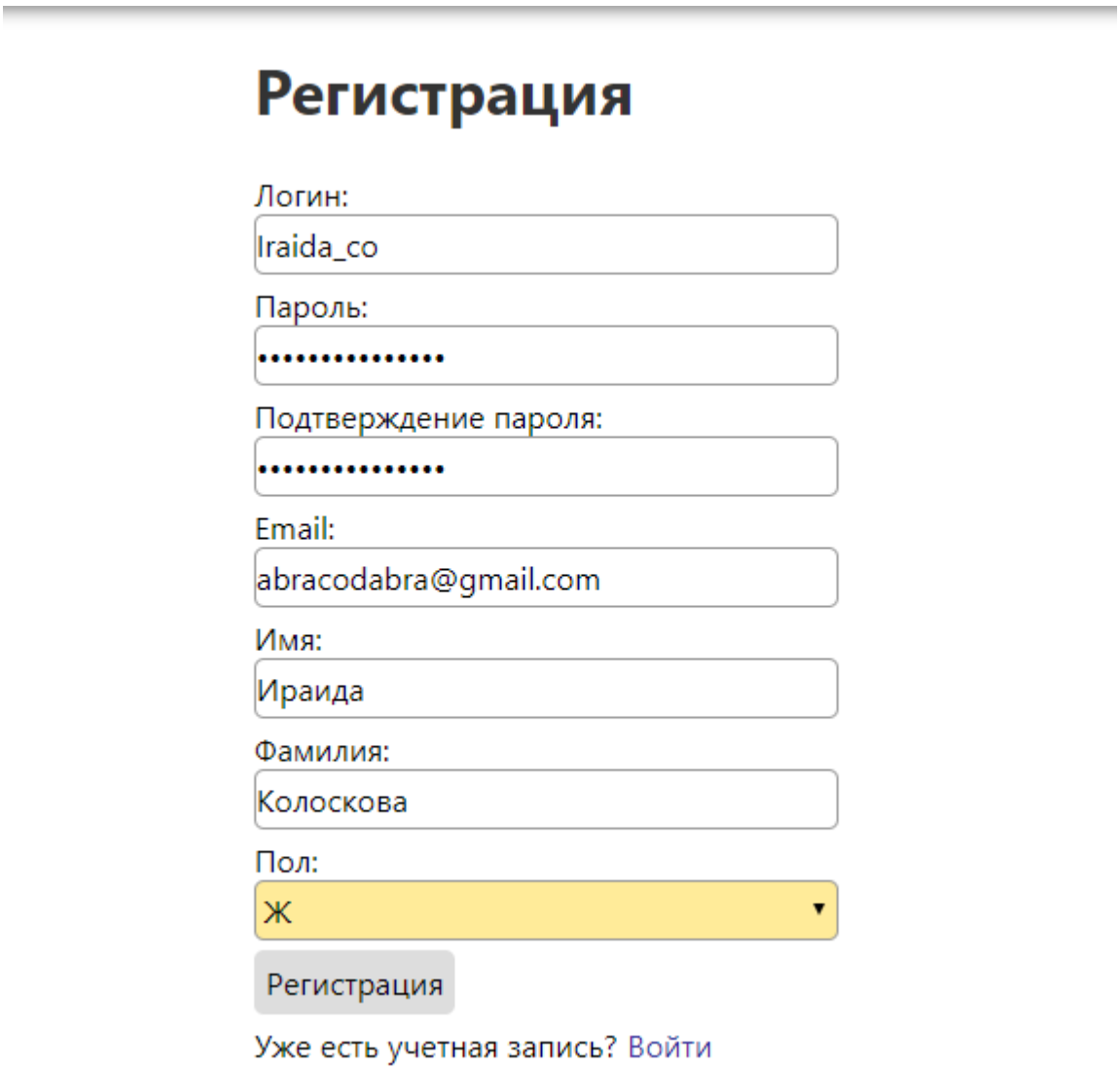


Рисунок 21 – Форма регистрации

После нажатия кнопки «Регистрация» на указанный адрес электронной почты приходит письмо от администратора.

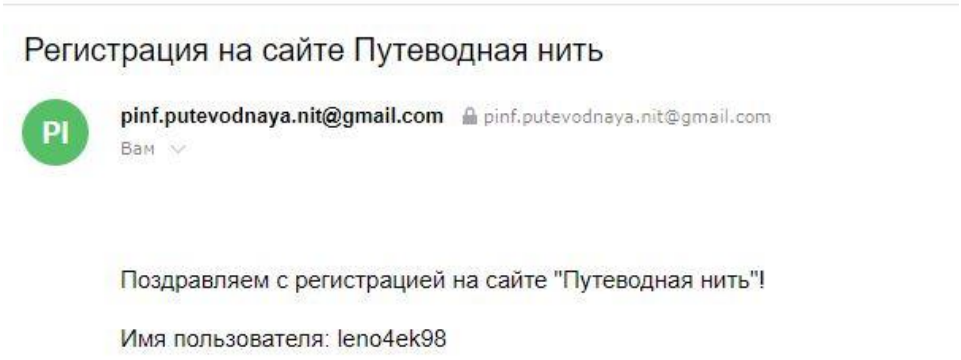


Рисунок 22 – Письмо с информацией о регистрации

После того, как администратор системы активирует учётную запись, на почту пользователя приходит ещё одно письмо (рис. 31).

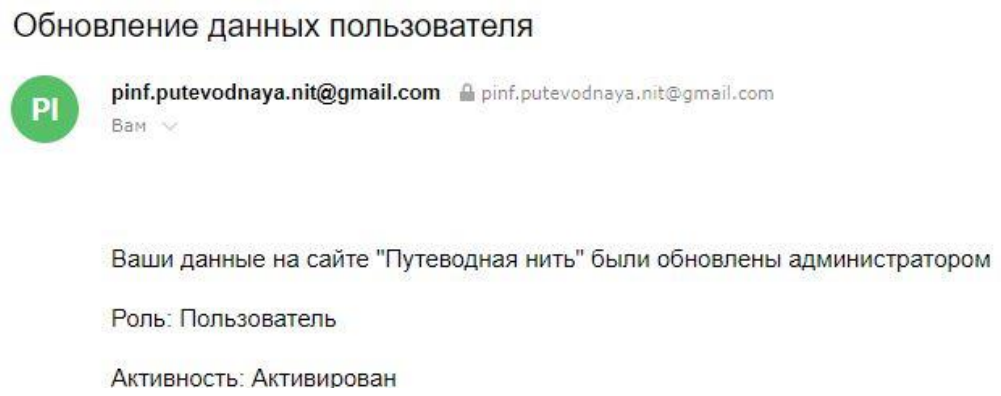


Рисунок 23 – Письмо с информацией об активации учётной записи

Теперь учётная запись пользователя отображается в панели администраторы, где администратор может активировать или деактивировать учётные записи, а также меняет роли, email, фамилии и имена пользователей.

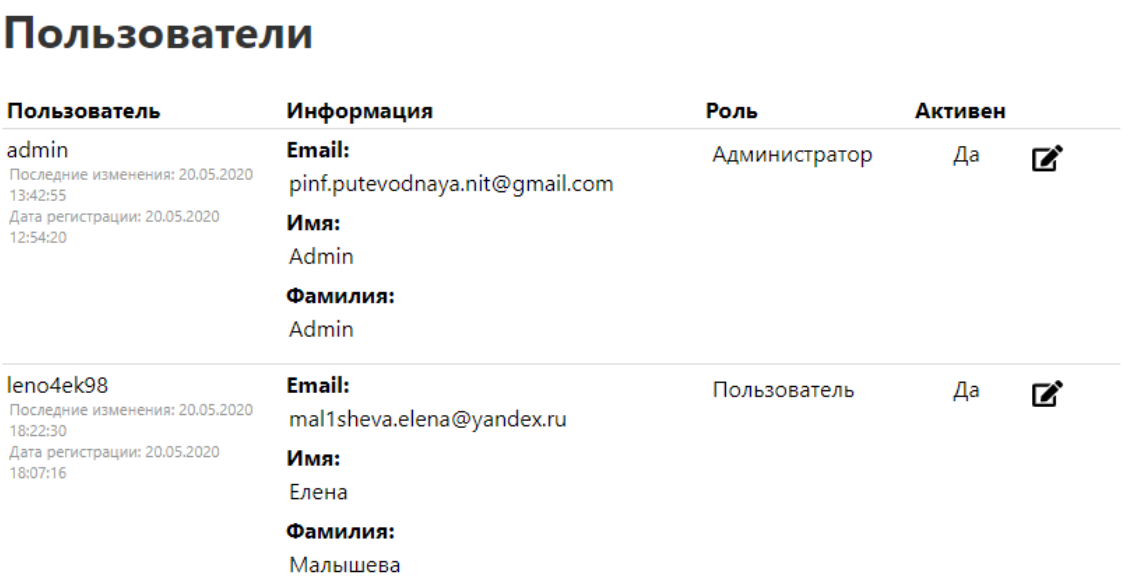


Рисунок 24 – Список пользователей системы

Каждый пользователь системы имеет личный кабинет, где может добавить или изменить аватар, увидеть параметры своей учетной записи (email, фамилию, имя, дату регистрации, дату последнего изменения профиля), редактировать пароль. При смене пароля пользователю также приходит оповещение на электронную почту.

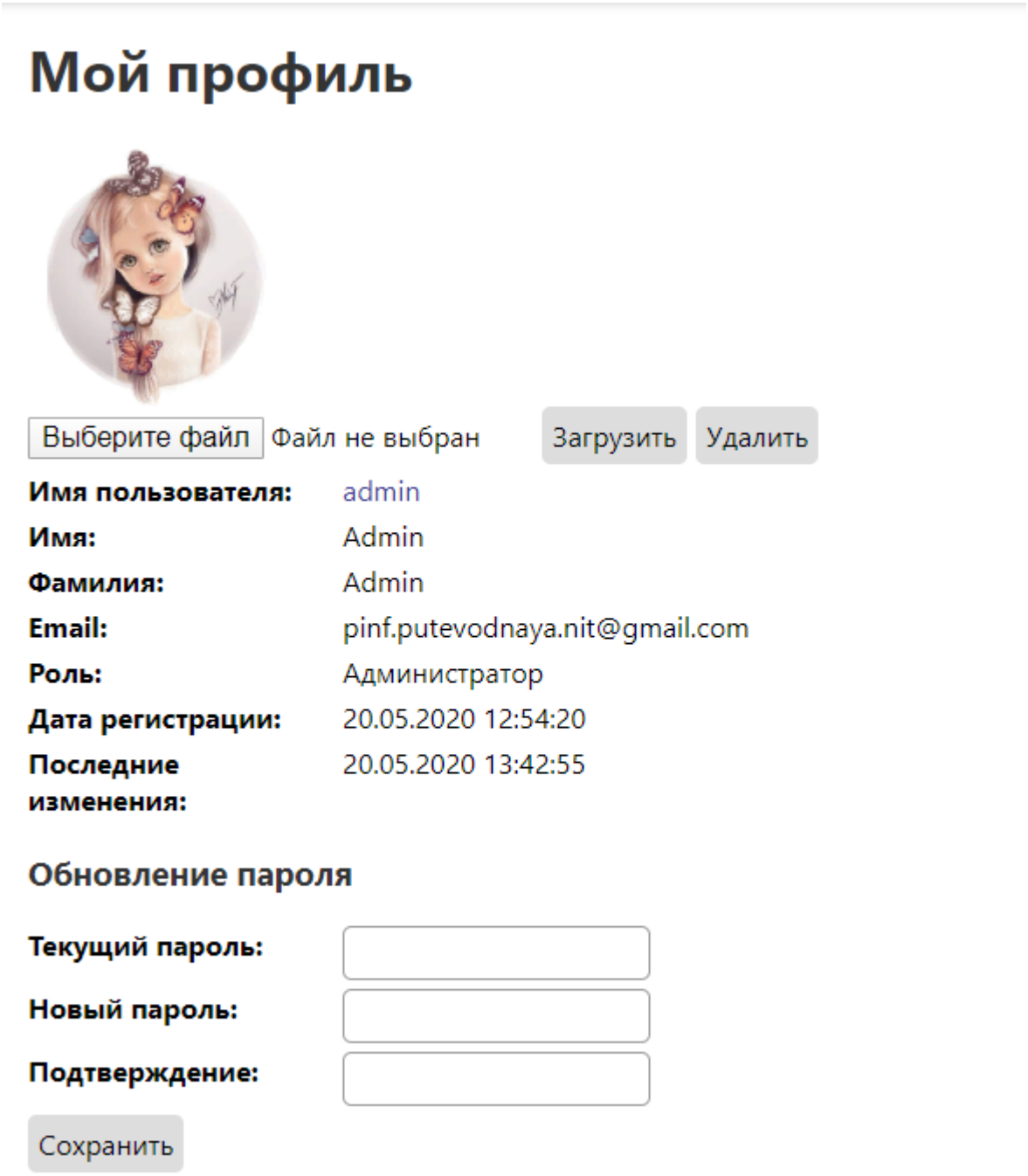


Рисунок 25 – Профиль пользователя

В разделе «Карта» при изменении масштаба включается кластеризация маркеров с целью оптимизации отображения большого количества маркеров. Несколько маркеров, находящихся в пределах небольшого радиуса, объединяются в кластер с указанием количества объединенных маркеров.

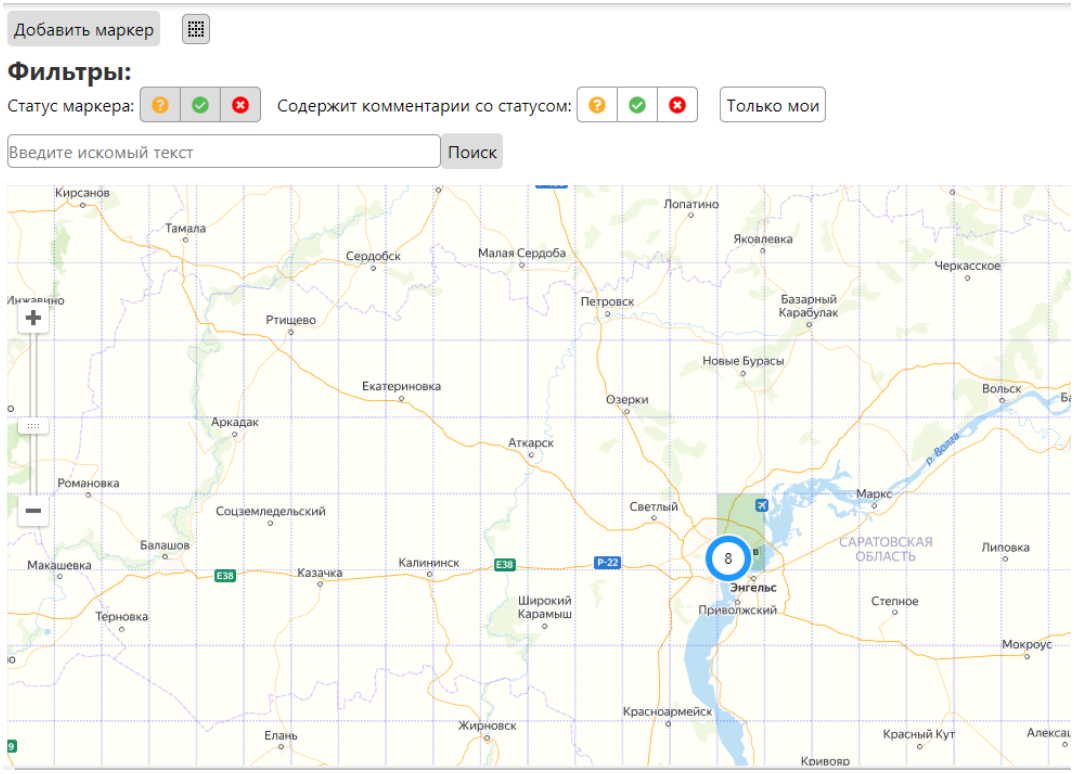


Рисунок 26 – Общее количество меток в городе

Также возможно увидеть общее количество добавленных в каждом районе маркеров.

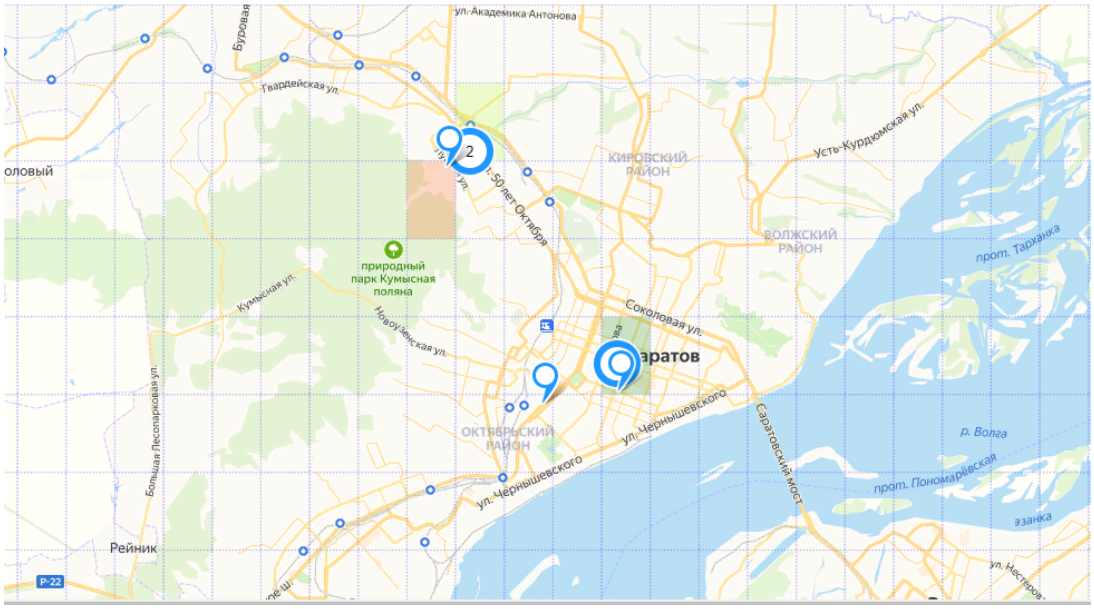


Рисунок 27 – Метки в каждом районе

Кроме того, пользователь может увидеть только собственные метки, нажав на кнопку «Только мои».

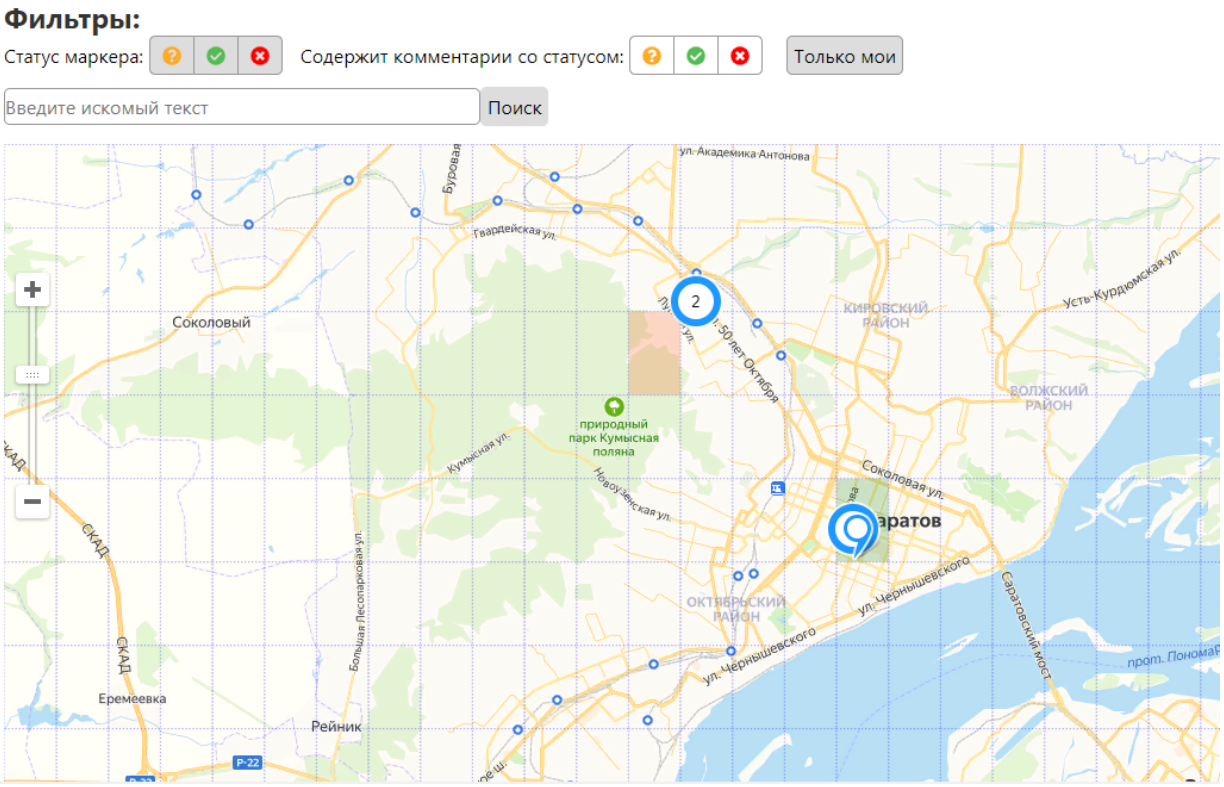


Рисунок 28 – Отфильтрованные метки на карте

Также метки можно отфильтровать по статусу доступности и части описания.

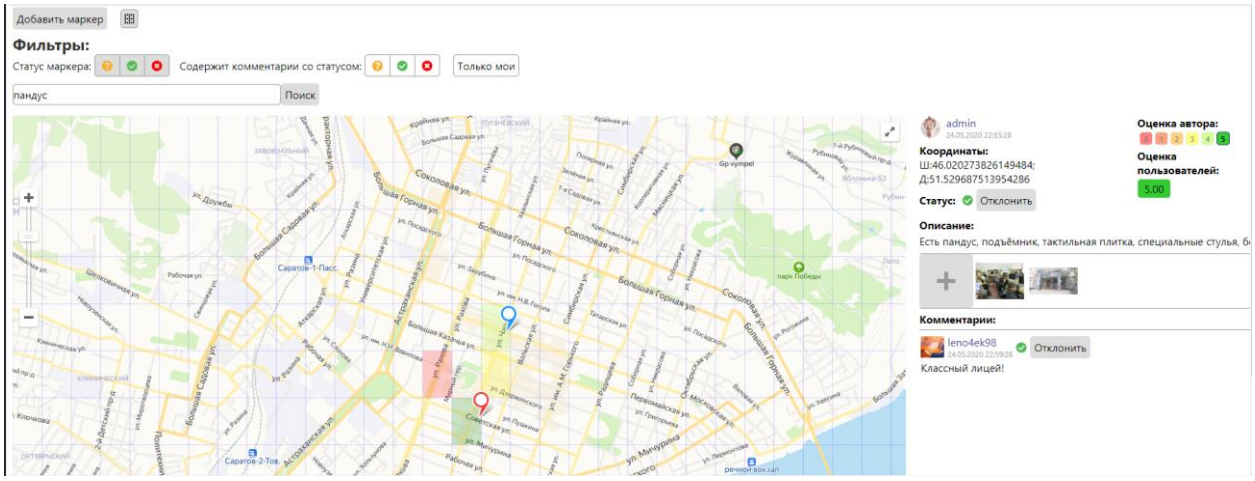


Рисунок 29 – Отфильтрованные по слову метки на карте

Модератор отклоняет метки, несоответствующие требованиям (например, содержащие некорректную информацию), и они не будут отображаться на карте для пользователей, имеющих роль «Пользователь» и не являющихся автором маркера.

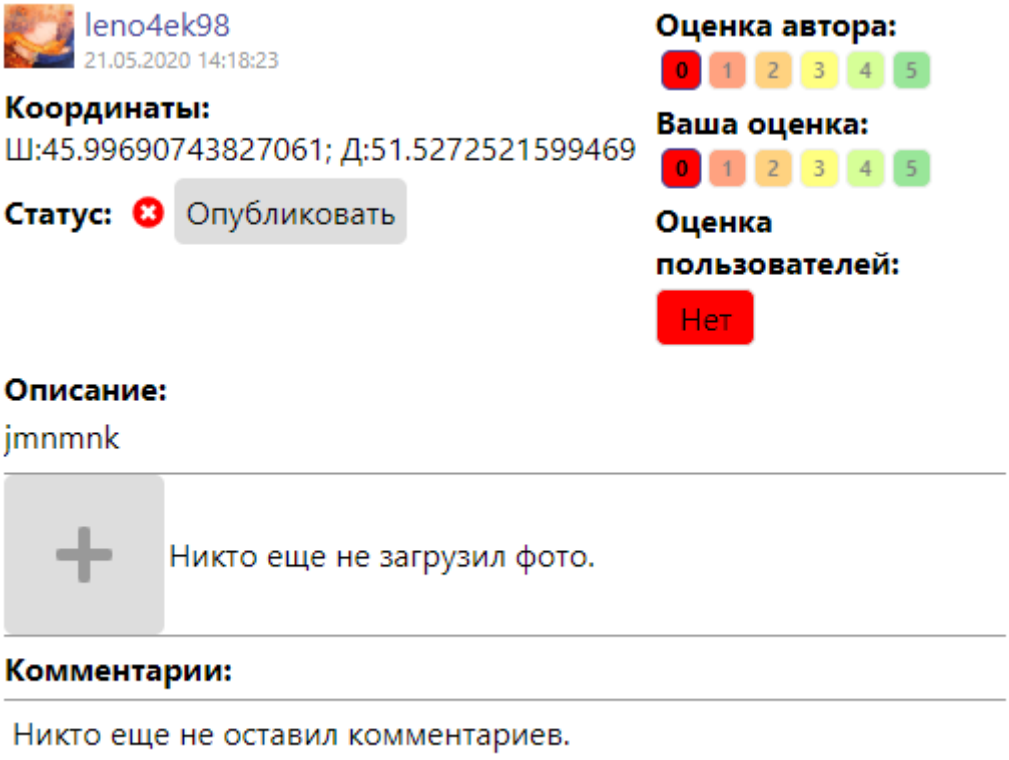


Рисунок 30 – Отклоненная метка

Карта делится на прямоугольники в соответствии с оценкой доступности объекта: темно-зелёный – доступен, жёлтый – частично доступен, красный – недоступен.

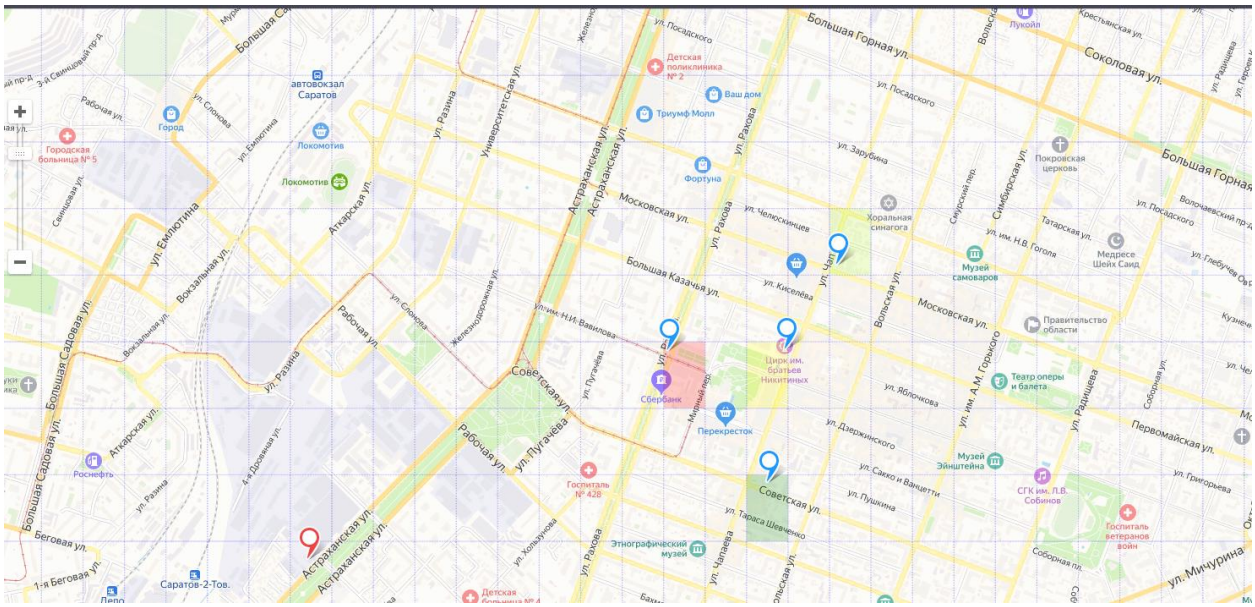


Рисунок 31 – Районирование города

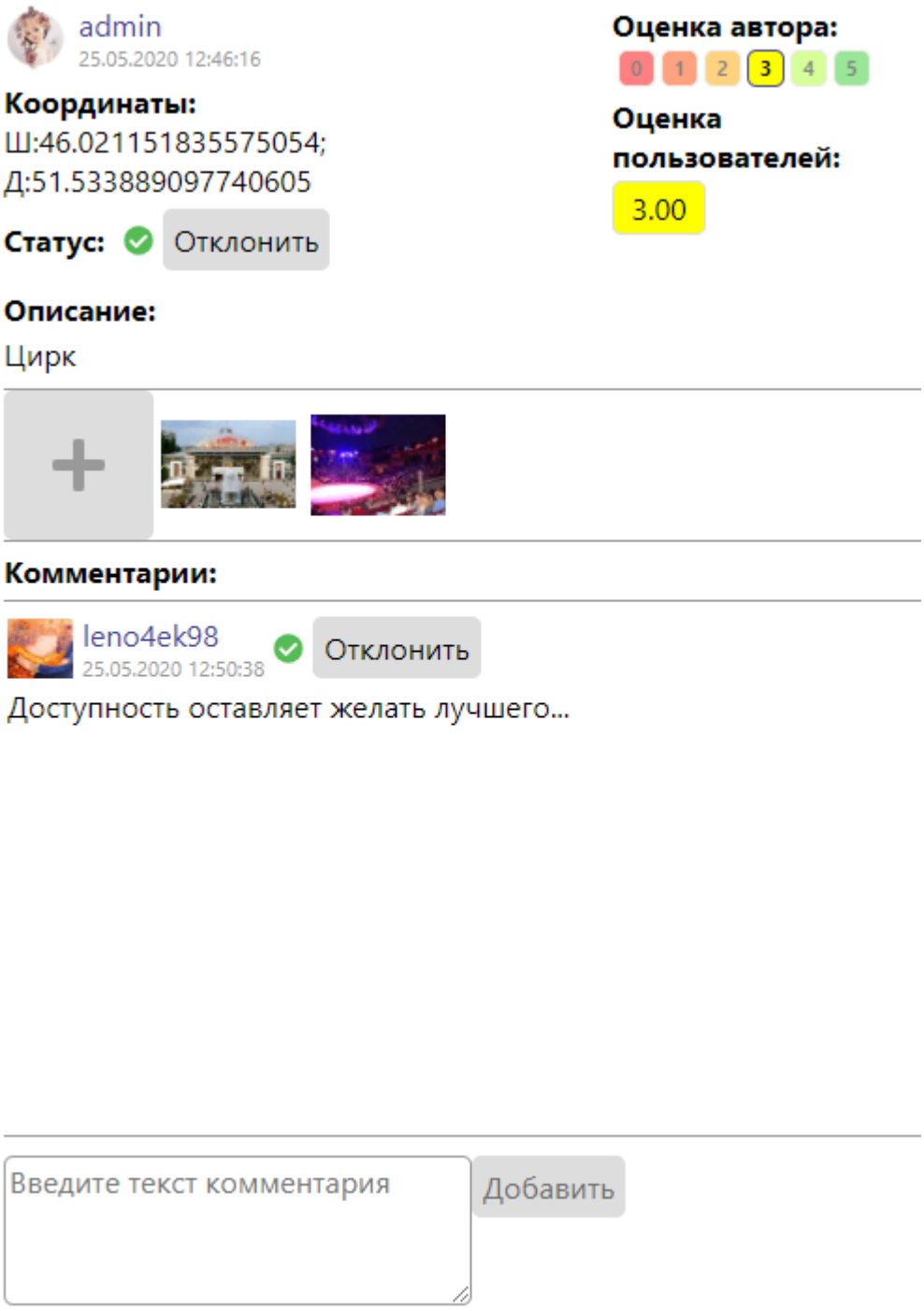


Рисунок 32 – Содержание маркера

Маркер содержит логин и аватар автора, дату и время добавления, координаты, статус с кнопками модерации (доступно только пользователям с ролью «Модератор» или «Администратор»), описание, список фотографий, список комментариев, оценку автора и усредненную оценку пользователей.

Пользователи могут загрузить любое количество фотографий с описанием.

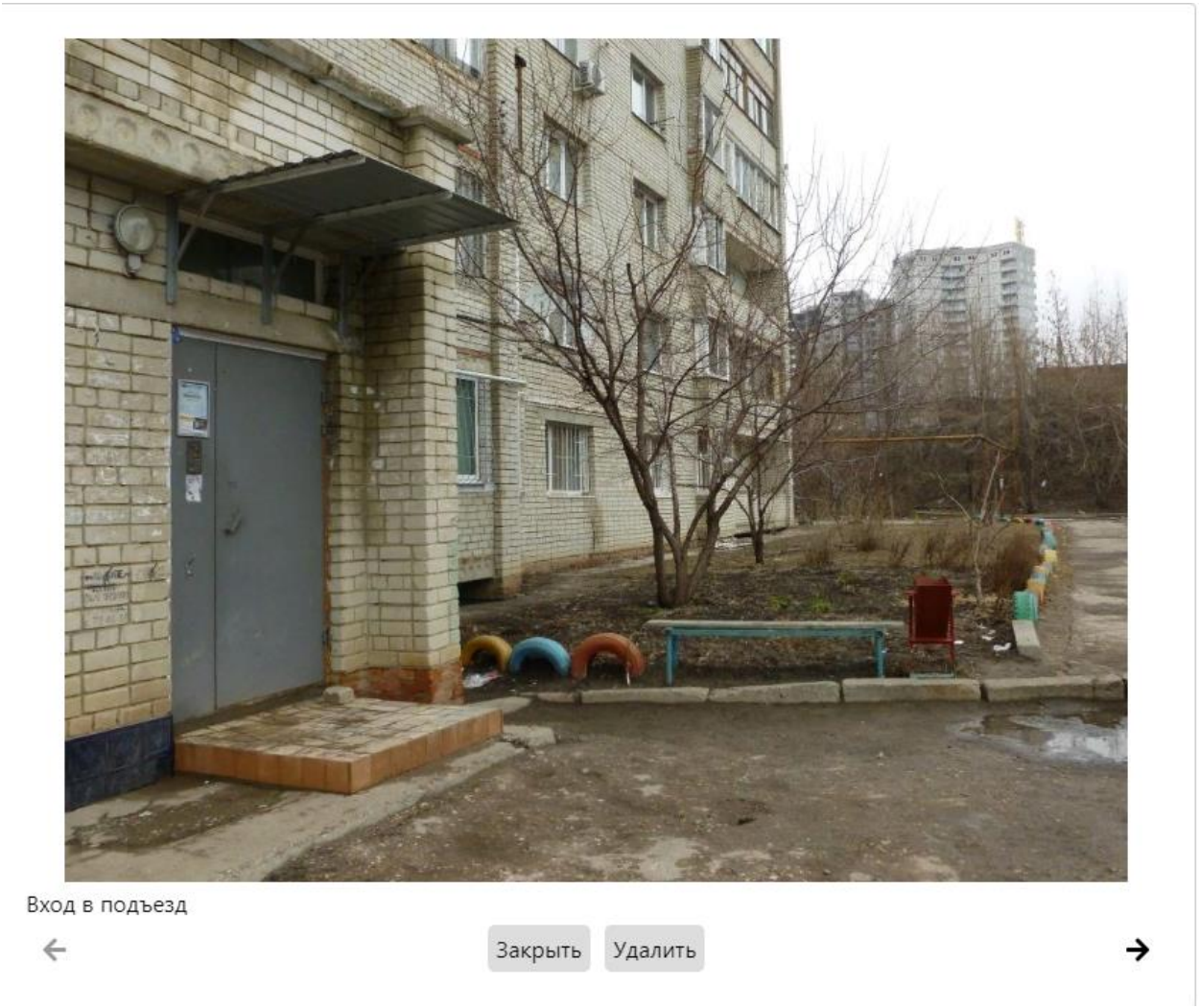


Рисунок 33 – Фотогалерея

Пользователи с ролью «Администратор» или «Модератор» могут удалять фотографии, не удовлетворяющие требованиям (например, не соответствующие реальному месту).

Для добавления метки нужно нажать кнопку «Добавить маркер», тем самым открыв форму добавления маркера. Далее необходимо указать место на карте, добавить описание и оценку. Для удобства пользователя карта на форме добавления маркера не содержит уже существующих маркеров.

После добавления метка не сразу появится на карте у автора, т.к. должна будет быть подтверждена пользователем с ролью «Модератор» или «Администратор», чтобы стать доступной остальным.

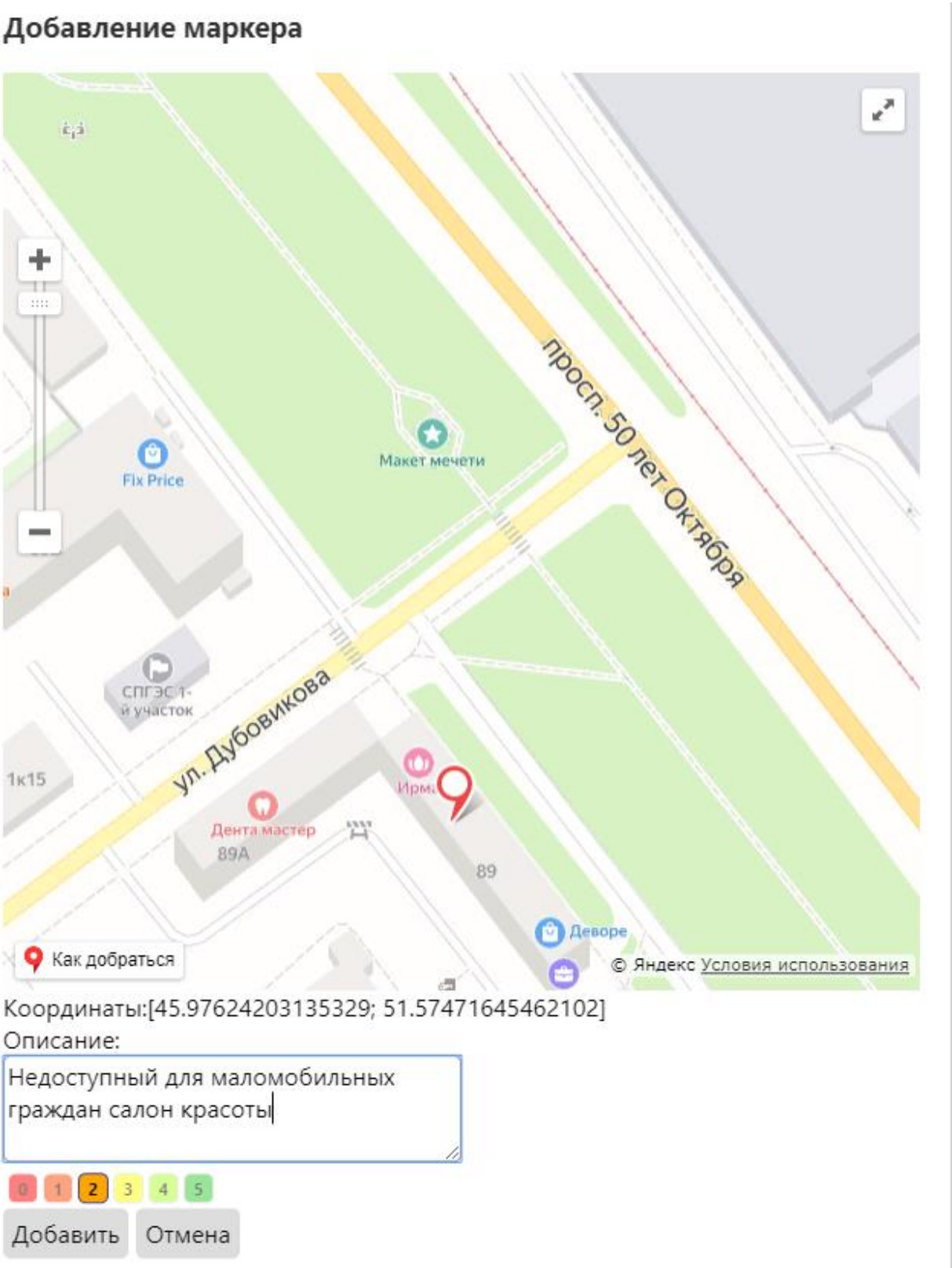


Рисунок 34 – Форма добавления маркера

Таким образом, в системе реализованы все запланированные функции, а поставленная цель работы достигнута.

# ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы была разработана геоинформационная система для маломобильных групп населения «Путеводная нить».

В разработанной геоинформационной системе был реализован следующий функционал:

- Добавление, удаление, редактирование меток (маркеров);
- Элементы социальной сети (фотографии, оценки, комментарии);
- Формирование карты доступности;
- Модерация меток и комментариев;
- Регистрация и авторизация пользователей.

В процессе разработки приложения были пройдены все необходимые этапы: от исследования предметной области до проверки работоспособности разработанного программного обеспечения. Разработанное программное обеспечение полностью удовлетворяет всем поставленным функциональным требованиям и выполнено в виде веб-приложения.

В качестве СУБД для разрабатываемой системы был выбран MySQL.

Для разработки серверной части использовались следующие инструменты: язык Java, Spring Framework, библиотеки Hibernate, Lombok, Mapstruct Mapstruct, JJWT, Jackson, Liquibase. Для разработки клиентской части использовались следующие инструменты: языки HTML, CSS, SASS, TypeScript, библиотеки React, Mobx, Axios, Webpack.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Тикунов В.С. Основы геоинформатики: В 2 кн.: учебное пособие для студентов вузов / М.: Издательский центр «Академия», 2004. – Кн. 1. – 352 с.
2. Тикунов В.С. Основы геоинформатики: В 2 кн.: учебное пособие для студентов вузов / М.: Издательский центр «Академия», 2004. – Кн. 2. – 480 с.
3. Вещикова Л.Ю., Сержевский Н.А. Проблема доступной среды // Инновационная экономика: перспектива развития и совершенствования. 2015. № 2 (7). С. 73-77.
4. Скрипкин, П. Б. Существующие проблемы доступной среды маломобильных групп населения в России и странах мира и мероприятия по их устранению / П. Б. Скрипкин, Р. С. Шаманов, Н. А. Михеева. — Текст: непосредственный // Молодой ученый. — 2014. — № 20 (79). — С. 217-220. — URL: https://moluch.ru/archive/79/14115/ (дата обращения: 26.05.2020).
5. Долинина О.Н., Печенкин В.В. О подходе к управлению сбором бытовых отходов с помощью гибридной интеллектуальной системы проекта "Умный город" // Программные системы и вычислительные методы. 2017. № 3. С. 1-15.
6. Неберушкина.Э.К. Перспективы создания доступной среды // Вестник СГТУ. 2012. № 1 (63) Выпуск 1. С. 205-208.
7. Жигунова Г.В. Обеспечение независимой жизни людей с инвалидностью в региональном социуме // Вестник университета. 2018. №12. С. 163 -169.
8. Аверина Е.А., Попова А.В. Оценка доступности среды для людей с ограниченными возможностями здоровья (на примере города Новосибирска) // Вестник Томского государственного университета. Философия. Социология. Политология. 2016. № 1 (33). С. 5 – 14.
9. ГИС «Доступная среда» [Электронный ресурс]. URL: https://samis.geosamara.ru/projects/detail.php?ID=4430 (дата обращения: 26.05.2020).

10. ДубльГИС: Нижний Новгород [Электронный ресурс]. URL: https://2gis.ru/n_novgorod (дата обращения: 26.05.2020).
11. Мобильное приложение для владельцев собак WOOOF [Электронный ресурс]. URL: http://www.sobaka.ru/lifestyle/gadgets/37411 (дата обращения: 26.05.2020).
12. Карта доступности объектов [Электронный ресурс]. URL: http://zhit-vmeste.ru/map/ (дата обращения: 26.05.2020).
13. Социальный навигатор [Электронный ресурс]. URL: http://sn.fss.ru/ (дата обращения: 26.05.2020).
14. MySql Documentation [Электронный ресурс]. URL: https://dev.mysql.com/doc/ (дата обращения: 26.05.2020).
15. Java Documentation [Электронный ресурс]. URL: https://docs.oracle.com/en/java/ (дата обращения: 26.05.2020).
16. Spring Boot Documentation [Электронный ресурс]. URL: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-documentation (дата обращения: 26.05.2020).
17. Hibernate ORM [Электронный ресурс]. URL: https://hibernate.org/orm/documentation/ (дата обращения: 26.05.2020).
18. Lombok features [Электронный ресурс]. URL: https://projectlombok.org/features/all (дата обращения: 26.05.2020).
19. Mapstruct Documentation [Электронный ресурс]. URL: https://mapstruct.org/documentation/ (дата обращения: 26.05.2020).
20. Introduction to JSON Web Tokens [Электронный ресурс]. URL: https://jwt.io/introduction/ (дата обращения: 26.05.2020).
21. Jackson API Tutorials [Электронный ресурс]. URL: https://www.concretepage.com/jackson-api/ (дата обращения: 26.05.2020).
22. Liquibase Online Help [Электронный ресурс]. URL: https://docsstage.liquibase.com/home.html (дата обращения: 26.05.2020).

23. Кириченко А., Хрусталев А. HTML5 + CSS3. Основы современного WEB-дизайна / Наука и Техника, 2018. - 354 с.
24. TypeScript Documentation [Электронный ресурс]. URL: https://www.typescriptlang.org/docs/home (дата обращения: 26.05.2020).
25. Руководство по TypeScript [Электронный ресурс]. URL: https://metanit.com/web/typescript/ (дата обращения: 26.05.2020).
26. MobX [Электронный ресурс]. URL: https://mobx.js.org/README.html (дата обращения: 26.05.2020).
27. Axios Documentation [Электронный ресурс]. URL: https://www.npmjs.com/package/axios (дата обращения: 26.05.2020).
28. SASS Документация на русском языке [Электронный ресурс]. URL: https://sass-scss.ru/ (дата обращения: 26.05.2020).
29. Шилдт Г. Java. Полное руководство / «Диалектика», 2018. – 1488 с.
30. Вайсфельд М. Объектно-ориентированное мышление / Питер, 2014. – 375 с.
31. Дюбуа П. MySQL. Сборник рецептов. /Символ-Плюс, 2007. - 1056 с.

## ПРИЛОЖЕНИЕ 1. Конфигурация клиентской части

```
{
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@babel/core": "7.9.0",
    "@fortawesome/fontawesome-free": "^5.13.0",
    "@svgr/webpack": "4.3.3",
    "@testing-library/jest-dom": "^4.2.4",
    "@testing-library/react": "^9.5.0",
    "@testing-library/user-event": "^7.2.1",
    "@types/jest": "^24.9.1",
    "@types/node": "^12.12.34",
    "@types/react": "^16.9.31",
    "@types/react-dom": "^16.9.6",
    "@typescript-eslint/eslint-plugin": "^2.10.0",
    "@typescript-eslint/parser": "^2.10.0",
    "autobind-decorator": "^2.4.0",
    "axios": "^0.19.2",
    "babel-eslint": "10.1.0",
    "babel-jest": "^24.9.0",
    "babel-loader": "8.1.0",
    "babel-plugin-named-asset-import": "^0.3.6",
    "babel-preset-react-app": "^9.1.2",
    "camelcase": "^5.3.1",
    "case-sensitive-paths-webpack-plugin": "2.3.0",
    "css-loader": "3.4.2",
    "dotenv": "8.2.0",
    "dotenv-expand": "5.1.0",
    "eslint": "^6.6.0",
    "eslint-config-react-app": "^5.2.1",
    "eslint-loader": "3.0.3",
    "eslint-plugin-flowtype": "4.6.0",
    "eslint-plugin-import": "2.20.1",
    "eslint-plugin-jsx-a11y": "6.2.3",
    "eslint-plugin-react": "7.19.0",
    "eslint-plugin-react-hooks": "^1.6.1",
    "file-loader": "4.3.0",
    "fs-extra": "^8.1.0",
    "html-webpack-plugin": "4.0.0-beta.11",
    "identity-obj-proxy": "3.0.0",
    "jest": "24.9.0",
    "jest-environment-jsdom-fourteen": "1.0.1",
    "jest-resolve": "24.9.0",
    "jest-watch-typeahead": "0.4.2",
    "mini-css-extract-plugin": "0.9.0",
    "mobx": "^5.15.4",
    "mobx-react": "^6.1.8",
    "optimize-css-assets-webpack-plugin": "5.0.3",
    "pnp-webpack-plugin": "1.6.4",
    "postcss-flexbugs-fixes": "4.1.0",
    "postcss-loader": "3.0.0",
    "postcss-normalize": "8.0.1",
    "postcss-preset-env": "6.7.0",
    "postcss-safe-parser": "4.0.1",
    "react": "^16.13.1",
    "react-app-polyfill": "^1.0.6",
    "react-dev-utils": "^10.2.1",
```

```
    "react-dom": "^16.13.1",
    "react-modal": "^3.11.2",
    "react-router-dom": "^5.1.2",
    "react-yandex-maps": "^4.2.0",
    "resolve": "1.15.0",
    "resolve-url-loader": "3.1.1",
    "semver": "6.3.0",
    "style-loader": "0.23.1",
    "terser-webpack-plugin": "2.3.5",
    "ts-pnp": "1.1.6",
    "typescript": "^3.8.3",
    "url-loader": "2.3.0",
    "webpack": "4.42.0",
    "webpack-dev-server": "3.10.3",
    "webpack-manifest-plugin": "2.2.0",
    "workbox-webpack-plugin": "4.3.1"
  },
  "scripts": {
    "start": "node scripts/start.js",
    "build": "node scripts/build.js",
    "test": "node scripts/test.js"
  },
  "eslintConfig": {
    "extends": "react-app"
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "jest": {
    "roots": [
      "<rootDir>/src"
    ],
    "collectCoverageFrom": [
      "src/**/*.{js,jsx,ts,tsx}",
      "!src/**/*.d.ts"
    ],
    "setupFiles": [
      "react-app-polyfill/jsdom"
    ],
    "setupFilesAfterEnv": [
      "<rootDir>/src/setupTests.ts"
    ],
    "testMatch": [
      "<rootDir>/src/**/__tests__/**/*.{js,jsx,ts,tsx}",
      "<rootDir>/src/**/*.{spec,test}.{js,jsx,ts,tsx}"
    ],
    "testEnvironment": "jest-environment-jsdom-fourteen",
    "transform": {
      "^.+\\.(js|jsx|ts|tsx)$": "<rootDir>/node_modules/babel-jest",
      "^.+\\.css$": "<rootDir>/config/jest/cssTransform.js",
      "^(?!.*\\.(js|jsx|ts|tsx|css|json)$)": "<rootDir>/config/jest/fileTransform.js"
```

```
    },
    "transformIgnorePatterns": [
      "[/\\\\]node_modules[/\\\\].+\\.(js|jsx|ts|tsx)$",
      "^.+\\.module\\.(css|sass|scss)$"
    ],
    "modulePaths": [],
    "moduleNameMapper": {
      "^react-native$": "react-native-web",
      "^.+\\.module\\.(css|sass|scss)$": "identity-obj-proxy"
    },
    "moduleFileExtensions": [
      "web.js",
      "js",
      "web.ts",
      "ts",
      "web.tsx",
      "tsx",
      "json",
      "web.jsx",
      "jsx",
      "node"
    ],
    "watchPlugins": [
      "jest-watch-typeahead/filename",
      "jest-watch-typeahead/testname"
    ]
  },
  "babel": {
    "presets": [
      "react-app"
    ]
  },
  "devDependencies": {
    "@types/react-modal": "^3.10.5",
    "@types/react-router-dom": "^5.1.3",
    "node-sass": "^4.13.1",
    "sass-loader": "^8.0.2"
  }
}
```

# ПРИЛОЖЕНИЕ 2. Конфигурация серверной части проекта

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.4.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.pn</groupId>
    <artifactId>pn</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>pn</name>
    <description>Geoinformation system</description>
    <packaging>jar</packaging>

    <properties>
        <file.encoding>UTF-8</file.encoding>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>1.8</java.version>
        <maven.compiler.target>1.8</maven.compiler.target>
        <maven.compiler.source>1.8</maven.compiler.source>

        <db.driver.class>com.mysql.cj.jdbc.Driver</db.driver.class>
        <db.url>jdbc:mysql://localhost/pn</db.url>
        <db.username>pn</db.username>
        <db.password>pnpass</db.password>

        <start-class>com.pn.app.PnApplication</start-class>

        <mapstruct.version>1.3.1.Final</mapstruct.version>
        <lombok.version>1.18.12</lombok.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-configuration-processor</artifactId>
            <optional>true</optional>
```

```
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
    <version>${lombok.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
    <exclusions>
        <exclusion>
            <groupId>org.junit.vintage</groupId>
            <artifactId>junit-vintage-engine</artifactId>
        </exclusion>
    </exclusions>
</dependency>

<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.19</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-jpamodelgen</artifactId>
</dependency>

<!-- Jackson for JSON generation -->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.10.2</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.10.2</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-annotations</artifactId>
    <version>2.10.2</version>
</dependency>

<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.1</version>
</dependency>

<dependency>
    <groupId>org.mapstruct</groupId>
    <artifactId>mapstruct</artifactId>
    <version>1.3.1.Final</version>
```

```
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-
starter-mail -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-mail</artifactId>
            <version>2.3.0.RELEASE</version>
        </dependency>


        <dependency>
            <groupId>com.sun.mail</groupId>
            <artifactId>javax.mail</artifactId>
            <version>1.6.2</version>
        </dependency>

        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.9.2</version>
        </dependency>

        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>2.9.2</version>
        </dependency>

    </dependencies>


    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <executable>true</executable>
                </configuration>
                <executions>
                    <execution>
                        <goals>
                            <goal>build-info</goal>
                        </goals>
                        <configuration>
                            <additionalProperties>
                                <encoding.source>UTF-8</encoding.source>
                                <encoding.reporting>UTF-8</encoding.reporting>
                                <java.source>${maven.compiler.source}</java.source>
                                <java.target>${maven.compiler.target}</java.target>
                            </additionalProperties>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.6.1</version>
```

```
                <configuration>
                    <source>${java.version}</source>
                    <target>${java.version}</target>
                    <annotationProcessorPaths>
                        <path>
                            <groupId>org.mapstruct</groupId>
                            <artifactId>mapstruct-processor</artifactId>
                            <version>${mapstruct.version}</version>
                        </path>
                        <path>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                            <version>${lombok.version}</version>
                        </path>
                    </annotationProcessorPaths>
                </configuration>
            </plugin>

            <plugin>
                <groupId>org.bsc.maven</groupId>
                <artifactId>maven-processor-plugin</artifactId>
                <version>2.2.4</version>
                <executions>
                    <execution>
                        <id>process</id>
                        <goals>
                            <goal>process</goal>
                        </goals>
                        <phase>generate-sources</phase>
                        <configuration>
                            <processors>
                                <proces-
sor>org.hibernate.jpamodelgen.JPAMetaModelEntityProcessor</processor>
                            </processors>
                        </configuration>
                    </execution>
                </executions>
                <dependencies>
                    <dependency>
                        <groupId>org.hibernate</groupId>
                        <artifactId>hibernate-jpamodelgen</artifactId>
                        <version>${hibernate.version}</version>
                    </dependency>
                </dependencies>
            </plugin>
        </plugins>


    </build>
    <profiles>
        <profile>
            <id>build-profile</id>
            <activation>
                <activeByDefault>true</activeByDefault>
            </activation>
            <build>
                <plugins>
                    <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-checkstyle-plugin</artifactId>
```

```
                    <version>3.1.1</version>
                    <configuration>
                        <configLocation>./checkstyle-checker.xml</configLocation>
                        <consoleOutput>true</consoleOutput>
                        <sourceDirecto-
ries>${project.build.sourceDirectory}</sourceDirectories>
                    </configuration>
                    <executions>
                        <execution>
                            <phase>compile</phase>
                            <goals>
                                <goal>check</goal>
                            </goals>
                        </execution>
                    </executions>
                </plugin>
                <plugin>
                    <groupId>org.liquibase</groupId>
                    <artifactId>liquibase-maven-plugin</artifactId>
                    <version>3.6.2</version>
                    <configuration>
                        <changeLogFile>./db/master.xml</changeLogFile>
                        <driver>${db.driver.class}</driver>
                        <url>${db.url}</url>
                        <username>${db.username}</username>
                        <password>${db.password}</password>
                    </configuration>
                    <executions>
                        <execution>
                            <phase>process-resources</phase>
                            <goals>
                                <goal>update</goal>
                            </goals>
                        </execution>
                    </executions>
                </plugin>
            </plugins>
        </build>
    </profile>
</profiles>
</project>
```

## ПРИЛОЖЕНИЕ 3. Программный код фильтра для поиска маркеров

```
@Slf4j
public class MarkerSpecification {
    /**
     * @param statuses        statuses
     * @param commentStatuses comment statuses
     * @param user            user
     * @param bounds          map bounds
     * @param searchText      search text
     * @param onlyMy          only my flag
     * @return specification
     */
    public static Specification<Marker> searchSpecification(
            List<StatusName> statuses,
            List<StatusName> commentStatuses,
            User user,
            MapBounds bounds,
            String searchText,
            Boolean onlyMy

    ) {
        Specification<Marker> spec = searchByBoundsSpecification(bounds);
        if (!CollectionUtils.isEmpty(commentStatuses)) {
            spec = spec.and(searchByCommentStatusSpecification(commentStatuses));
        }
        Specification<Marker> userSpec = searchByUserSpecification(user);
        if (Boolean.TRUE.equals(onlyMy)) {
            spec = spec.and(userSpec);
        }

        if (!CollectionUtils.isEmpty(statuses)) {
            spec = spec.and(searchByStatusSpecification(statuses));
        }
        if (!StringUtils.isEmpty(searchText)) {
            spec = spec.and(searchByTextSpecification(searchText));
        }
        if (UserRole.ROLE_USER.equals(user.getRole())) {
            Specification<Marker> othersApprovedSpec =
                    Specifica-
tion.not(userSpec).and(searchByStatusSpecification(Arrays.asList(StatusName.APPROVED)
));
            spec = spec.and(othersApprovedSpec.or(userSpec));
        }
        return spec;
    }

    /**
     * @param searchText search text
     * @return specification
     */
    public static Specification<Marker> searchByTextSpecification(String searchText)
{
        return Specification.where((root, criteriaQuery, criteriaBuilder) ->
                criteriaBuilder.like(
                        criteriaBuilder.lower(root.get(Marker_.description)),
                        "%" + searchText.toLowerCase() + "%"
                )
        );
    }
```

```
    /**
     * @param statuses statuses
     * @return specification
     */
    public static Specification<Marker> searchByStatusSpecification(List<StatusName>
statuses) {
        return Specification.where((root, criteriaQuery, criteriaBuilder) -> root
                .join(Marker_.status)
                .get(Status_.name)
                .in(statuses)
        );
    }

    /**
     * @param statuses statuses
     * @return specification
     */
    public static Specification<Marker> searchByCommentStatusSpecifica-
tion(List<StatusName> statuses) {
        return Specification.where((r, cq, cb) -> {
                    final ListJoin<Marker, Comment> commentJoin =
r.join(Marker_.comments);
                    cq.groupBy(
                            r.get(Marker_.id)
                    );
                    return commentJoin
                            .join(Comment_.status)
                            .get(Status_.name)
                            .in(statuses);
                }
        );
    }

    /**
     * @param user user
     * @return specification
     */
    public static Specification<Marker> searchByUserSpecification(User user) {
        return Specification.where(
                (root, criteriaQuery, criteriaBuilder) ->
root.join(Marker_.user).get(User_.id).in(user.getId())
        );
    }

    /**
     * @param bounds map bounds
     * @return specification
     */
    public static Specification<Marker> searchByBoundsSpecification(
            MapBounds bounds) {
        final Specification<Marker> latSpec =
                Specification.where(
                        (root, cq, cb) -> cb.between(
                                root.get(Marker_.latitude),
                                bounds.getLatBottomLeft(),
                                bounds.getLatTopRight()
                        )
                );
        final Specification<Marker> lngSpec =
```

```
                Specification.where(
                        (root, cq, cb) -> cb.between(
                                root.get(Marker_.longitude),
                                bounds.getLngBottomLeft(),
                                bounds.getLngTopRight()
                        )
                );
        return latSpec.and(lngSpec);
    }
}
```

## ПРИЛОЖЕНИЕ 4. Программный код класса сервиса для работы с маркерами

```
@Slf4j
@Service
public class MarkerService extends EntityService<Marker> implements EntityToUserLink-
Checker<Marker> {

    @Autowired
    private MarkerRepository markerRepository;

    @Autowired
    private UserService userService;

    @Autowired
    private StatusService statusService;

    /**
     * @return marker repository
     */
    @Override
    BaseRepository<Marker> getRepository() {
        return markerRepository;
    }

    /**
     * @return created marker
     */
    @Override
    Marker create() {
        return new Marker();
    }

    /**
     * @param latitude    latitude
     * @param longitude   longitude
     * @param owner       owner
     * @param description description
     * @param mark        mark
     * @return marker
     */
    public Marker add(
            @NonNull Double latitude,
            @NonNull Double longitude,
            @NonNull User owner,
            @NonNull String description,
            @NonNull Integer mark
    ) {
        Marker m = create();
        m.setLatitude(latitude);
        m.setLongitude(longitude);
        m.setUser(owner);
        m.setDescription(description);
        m.setStatus(statusService.getDefaultStatus());
        m.setMark(mark);
        return getRepository().save(m);
    }

    /**
```

```
     * @param latitude    latitude
     * @param longitude   longitude
     * @param description description
     * @param mark        mark
     * @return added marker
     */
    public Marker addForCurrentUser(
            @NonNull Double latitude,
            @NonNull Double longitude,
            @NonNull String description,
            @NonNull Integer mark
    ) {
        return add(latitude, longitude, userService.getCurrentUser(), description,
mark);
    }

    /**
     * @param user user
     * @return list of markers
     */
    public List<Marker> findAllByUser(@NonNull User user) {
        return markerRepository.findAllByUser(user);
    }

    /**
     * @return list of markers
     */
    public List<Marker> findAllForCurrentUser() {
        return findAllByUser(userService.getCurrentUser());
    }

    /**
     * @param statuses        statuses
     * @param commentStatuses comment statuses
     * @param onlyMy          only my flag
     * @param bounds          map bounds
     * @param searchText      search text
     * @return markers
     */
    public List<Marker> findAll(
            List<StatusName> statuses, List<StatusName> commentStatuses,
            Boolean onlyMy, MapBounds bounds, String searchText

    ) {
        User currentUser = userService.getCurrentUser();
        if (UserRole.ROLE_USER.equals(currentUser.getRole())) {
            statuses = null;
        }
        return markerRepository.findAll(
                MarkerSpecification.searchSpecification(
                        statuses, commentStatuses, currentUser, bounds, searchText,
onlyMy
                )
        );
    }


    /**
     * @param id marker ID
     */
```

```
    public void deleteForCurrentUser(@NonNull Integer id) {
        final Optional<Marker> optionalMarker = findById(id);
        if (optionalMarker.isPresent()) {
            final Marker marker = optionalMarker.get();
            deleteForCurrentUser(marker);
        } else {
            throw new EntityNotFoundException(id);
        }
    }

    /**
     * @param marker marker
     */
    public void deleteForCurrentUser(@NonNull Marker marker) {
        if (isLinked(marker, userService.getCurrentUser())) {
            delete(marker);
        }
        throw new AccessViolationException();
    }

    @Override
    public final boolean isLinked(@NonNull final Marker entity, @NonNull final User
user) {
        return entity.getUser().equals(user);
    }

    /**
     * @param marker     marker
     * @param statusName status name
     * @return updated comment
     */
    public Marker changeStatus(@NonNull Marker marker, @NonNull StatusName status-
Name) {
        if (!statusName.equals(marker.getStatus().getName())) {
            final Status newStatus = statusService.findByName(statusName);
            marker.setStatus(newStatus);
            return save(marker);
        }
        return marker;
    }


    /**
     * @param bounds map bounds
     * @return rectangles
     */
    public List<MapRectangle> getRectangles(
            MapBounds bounds
    ) {
        final List<MarkerOnMap> markers = markerRepository.findForRatingsMap(
                bounds.getLatBottomLeft(), bounds.getLngBottomLeft(),
bounds.getLatTopRight(), bounds.getLngTopRight()
        );

        double latDiff = bounds.getLatTopRight() - bounds.getLatBottomLeft();
        double lngDiff = bounds.getLngTopRight() - bounds.getLngBottomLeft();
        double minSize = Math.min(latDiff, lngDiff);
        double step = minSize / NumericConstants.TEN;
        int rectHorizontally = (int) Math.ceil(latDiff / step) + 1;
        int rectVertically = (int) Math.ceil(lngDiff / step) + 1;
```

```
        List<List<MapRectangle>> rectMatrix = new ArrayList<>(rectVertically);
        for (int i = 0; i < rectVertically; i++) {
            List<MapRectangle> row = new ArrayList<>(rectHorizontally);
            rectMatrix.add(row);
            for (int j = 0; j < rectHorizontally; j++) {
                MapRectangle r = new MapRectangle(
                        bounds.getLatBottomLeft() + j * step,
                        bounds.getLngBottomLeft() + i * step,
                        bounds.getLatBottomLeft() + (j + 1) * step,
                        bounds.getLngBottomLeft() + (i + 1) * step,
                        null,
                        0
                );
                row.add(r);
            }
        }

        for (MarkerOnMap m : markers) {
            int rectCol = (int) Math.floor((m.getLatitude() -
bounds.getLatBottomLeft()) / step);
            int rectRow = (int) Math.floor((m.getLongitude() -
bounds.getLngBottomLeft()) / step);
            MapRectangle rectangle = rectMatrix.get(rectRow).get(rectCol);
            if (m.getAverageMark() != null) {
                rectangle.setCount(rectangle.getCount() + 1);
                if (rectangle.getValue() == null) {
                    rectangle.setValue(0D);
                }
                rectangle.setValue(rectangle.getValue() + m.getAverageMark());
            }
        }
        return rectMa-
trix.stream().flatMap(Collection::stream).collect(Collectors.toList());
    }
}
```

## ПРИЛОЖЕНИЕ 5. Программный код класса сервиса для работы с JWT

```
@Slf4j
@Service
public class JwtService {
    public static final int TOKEN_TTL = DateTimeUtil.MILLISECONDS_IN_HOUR * 7;
    @Value("${security.jwt.secret}")
    private String secret;

    /**
     * @param token JWT
     * @return username
     */
    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }

    /**
     * @param token JWT
     * @return username
     */
    public Date extractExpiration(String token) {
        return extractClaim(token, Claims::getExpiration);
    }

    /**
     * @param token          JWT
     * @param claimResolver claim resolving function
     * @param <T>            claim type
     * @return claim value
     */
    public <T> T extractClaim(String token, Function<Claims, T> claimResolver) {
        Claims claims = extractAllClaims(token);
        return claimResolver.apply(claims);

    }

    /**
     * @param token JWT
     * @return all claims
     */
    private Claims extractAllClaims(String token) {
        return Jwts.parser().setSigningKey(secret).parseClaimsJws(token).getBody();
    }

    /**
     * @param token JWT
     * @return is expired
     */
    private boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());
    }

    /**
     * @param userDetails user details
     * @return JWT
     */
    public String generateToken(UserDetails userDetails) {
        Map<String, Object> claims = new HashMap<>();
        return generateToken(claims, userDetails.getUsername());
```

```
    }

    /**
     * @param claims  claims
     * @param subject subject
     * @return JWT
     */
    private String generateToken(Map<String, Object> claims, String subject) {
        final long millis = System.currentTimeMillis();
        return Jwts.builder()
                .setClaims(claims)
                .setSubject(subject)
                .setIssuedAt(new Date(millis))
                .setExpiration(new Date(millis + TOKEN_TTL))
                .signWith(SignatureAlgorithm.HS256, secret).compact();
    }

    /**
     * @param token       JWT
     * @param userDetails user details
     * @return is valid
     */
    public Boolean validateToken(String token, UserDetails userDetails) {
        String username = extractUsername(token);
        return (username.equals(userDetails.getUsername()) && !isTokenEx-
pired(token));
    }
}
```